TECHNISCHE UNIVERSITÄT DRESDEN

Recovering the Semantics of Tabular Web Data

Dissertation

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dipl.Medien-Inf. Katrin Braunschweig
geboren am 28. August 1984 in Suhl

Gutachter:

Prof. Dr.-Ing. Wolfgang Lehner
Technische Universität Dresden
Fakultät Informatik, Institut für Systemarchitektur
Lehrstuhl für Datenbanken
01062 Dresden

Prof. Dr. Stefan Conrad
Heinrich-Heine-Universität Düsseldorf
Mathematisch-Naturwissenschaftliche Fakultät, Institut für Informatik
Lehrstuhl für Datenbanken und Informationssysteme
40225 Düsseldorf

Tag der Verteidigung: 09. Oktober 2015

ABSTRACT

The Web provides a platform for people to share their data, leading to an abundance of accessible information. In recent years, significant research effort has been directed especially at *tables* on the Web, which form a rich resource for factual and relational data. Applications such as fact search and knowledge base construction benefit from this data, as it is often less ambiguous than unstructured text. However, many traditional information extraction and retrieval techniques are not well suited for Web tables, as they generally do not consider the role of the table structure in reflecting the semantics of the content. Tables provide a compact representation of similarly structured data. Yet, on the Web, tables are very heterogeneous, often with ambiguous semantics and inconsistencies in the quality of the data. Consequently, recognizing the structure and inferring the semantics of these tables is a challenging task that requires a designated *table recovery* and *understanding* process.

In the literature, many important contributions have been made to implement such a table understanding process that specifically targets Web tables, addressing tasks such as table detection or header recovery. However, the precision and coverage of the data extracted from Web tables is often still quite limited. Due to the complexity of Web table understanding, many techniques developed so far make simplifying assumptions about the table layout or content to limit the amount of contributing factors that must be considered. Thanks to these assumptions, many subtasks become manageable. However, the resulting algorithms and techniques often have a limited scope, leading to imprecise or inaccurate results when applied to tables that do not conform to these assumptions.

In this thesis, our objective is to extend the Web table understanding process with techniques that enable some of these assumptions to be relaxed, thus improving the scope and accuracy. We have conducted a comprehensive analysis of tables available on the Web to examine the characteristic features of these tables, but also identify unique challenges that arise from these characteristics in the table understanding process. To extend the scope of the table understanding process, we introduce extensions to the subtasks of table classification and conceptualization. First, we review various table layouts and evaluate alternative approaches to incorporate layout classification into the process. Instead of assuming a single, uniform layout across all tables, recognizing different table layouts enables a wide range of tables to be analyzed in a more accurate and systematic fashion. In addition to the layout, we also consider the conceptual level. To relax the *single concept assumption*, which expects all attributes in a table to describe the same semantic concept, we propose a semantic normalization approach. By decomposing multi-concept tables into several single-concept tables, we further extend the range of Web tables that can be processed correctly, enabling existing techniques to be applied without significant changes.

Furthermore, we address the quality of data extracted from Web tables, by studying the role of context information. Supplementary information from the context is often required to correctly understand the table content, however, the verbosity of the surrounding text can also mislead any table relevance decisions. We first propose a selection algorithm to evaluate the relevance of context information with respect to the table content in order to reduce the noise. Then, we introduce a set of extraction techniques to recover attribute-specific information from the relevant context in order to provide a richer description of the table content.

With the extensions proposed in this thesis, we increase the scope and accuracy of Web table understanding, leading to a better utilization of the information contained in tables on the Web.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the help and support of many colleagues and friends.
First, I must thank my advisor Wolfgang Lehner, who introduced me to database research and opened up a world of opportunities when he offered me to join his group at Technische Universität Dresden in 2010. Throughout the years, he provided many valuable comments on my work and offered his advice when needed. From the start, Wolfgang trusted in my abilities, which greatly encouraged me in my research, and I am very grateful for his support.
I owe special thanks to my fellow EDYRA researchers, Maik Thiele and Julian Eberius, who both had a significant impact on my work. As my co-advisor, Maik has guided me in my research throughout the years and always supported me through his advice, encouragements and patience during seemingly endless discussions about my work. Julian has collaborated with me on many projects and throughout the last couple of years, we often shared the ups and downs of graduate research. I am deeply grateful for their support and friendship.
Furthermore, I would like to thank Dirk Habich and Steffen Preißler. They have both been very generous with their time, providing support and offering advice whenever it was needed.
I would also like to acknowledge the contributions my students have made to the work presented here, often taking on the laborious task of manually labeling data for evaluation.
In addition, I would like to express my gratitude to Professor Stefan Conrad for taking the time to co-referee this thesis, despite the tight schedule.
During this research, I have been very fortunate to work with a fantastic group of people who have made this such a positive experience for me. I will never forget the many fun times we had both in the office and after work. Many thanks to all my colleagues at the database group for their support.
I would also like to thank my friends for their constant support, offering kind words of encouragement or a chance for distraction, depending on what was needed most. Last but not least, I owe the most gratitude to my family. I would not be able to pursue my dreams without their love and support.

Katrin Braunschweig
Dresden, October 21, 2015

CONTENTS

1 Introduction 11

1.1 Motivation . . . 12

1.2 Contributions . . . 14

1.3 Outline . . . 15

2 Foundations of Tables and Table Understanding 17

2.1 Tables and Relations . . . 18

2.1.1 Tables in Documents . . . 19

2.1.2 Tables in Relational Databases . . . 20

2.1.3 Linking Document and Database Tables . . . 21

2.2 Automated Table Recovery . . . 23

2.2.1 Process Overview . . . 23

2.2.2 Survey of Related Work . . . 24

2.2.3 Application to Tables on the Web . . . 25

3 Web Table Understanding 27

3.1 Tables on the Web . . . 28

3.1.1 File Formats . . . 29

3.1.2 Table Layout . . . 30

3.1.3 Data Quality and Attribute Semantics . . . 32

3.1.4 Table Size . . . 33

3.1.5 Corpus Size and Domain Coverage . . . 34

3.2 Web Table Applications . . . 36

3.2.1 Knowledge Extraction and Ontology Learning . . . 36

3.2.2 Question Answering . . . 38

3.2.3 Entity Augmentation and Situational Data Analysis . . . 40

3.2.4 Analysis of Application Requirements . . . 42
3.3 Revisiting Table Recovery . . . 42
3.3.1 Table Recognition . . . 43
3.3.2 Table Understanding . . . 43
3.3.3 Simplifying Assumptions . . . 48
3.3.4 Outline . . . 49
4 Table Extraction and Classification 51
4.1 Related Work . . . 53
4.2 Classification Problem . . . 57
4.3 Feature Specification . . . 58
4.3.1 Table Features . . . 58
4.3.2 Subset Features . . . 61
4.3.3 Pre-Selection Filters . . . 63
4.4 Experimental Evaluation . . . 63
4.4.1 Dataset . . . 63
4.4.2 Feature Selection . . . 64
4.4.3 Classifiers . . . 65
4.4.4 Evaluation . . . 67
4.5 Summary and Discussion . . . 72
5 Recovering Web Table Context 75
5.1 Web Table Context . . . 77
5.2 Related Work . . . 79
5.3 Context Relevance . . . 80
5.3.1 Problem Statement . . . 82
5.3.2 Text Segmentation . . . 83
5.3.3 Word-based Similarity . . . 85
5.3.4 Topic-based Similarity . . . 91
5.3.5 Filter and Threshold - Context Selection . . . 94
5.3.6 Discussion . . . 96
5.4 Attribute-specific Context Annotation . . . 96
5.4.1 Problem Statement . . . 97

5.4.2 Directly Related Context . . . 97
5.4.3 Indirectly Related Context . . . 99
5.5 Experimental Evaluation . . . 103
5.5.1 Experimental Setup . . . 103
5.5.2 Column-specific Context . . . 104
5.5.3 Context in Search Applications . . . 106
5.6 Summary and Discussion . . . 109
6 Semantic Normalization 111
6.1 Relation to Database Normalization . . . 113
6.1.1 Foundations of Database Normalization . . . 114
6.1.2 Related Work . . . 115
6.2 Formalization of Semantic Normalization . . . 118
6.3 Challenges . . . 119
6.4 Normalization Process . . . 122
6.4.1 Entity Column Identification . . . 123
6.4.2 Candidate Dependency Extraction . . . 127
6.4.3 Indicators for Attribute Relatedness . . . 130
6.4.4 Dependency Filtering and Decomposition . . . 134
6.4.5 Optimization: Table Clustering . . . 137
6.5 Experimental Evaluation . . . 138
6.5.1 Entity Columns . . . 140
6.5.2 Functional Dependency Extraction and Filtering . . . 141
6.5.3 End-To-End Processing . . . 144
6.5.4 Clustering . . . 145
6.6 Summary and Discussion . . . 146
7 Conclusion 149
7.1 Summary of Contributions . . . 150
7.2 Directions of Future Work . . . 151
A Layouts of Tables on the Web 153
A.1 Vertical Listing . . . 154

A.2 Horizontal Listing 154
A.3 Matrix 155
A.4 Special Cases 155

B Corpora 157

B.1 Dresden Web Table Corpus 158
B.2 Wikipedia Corpus 158

C Web Table Context 161

C.1 Embedded Web Tables 162
C.2 Open Data Tables 163

D Evaluation of Context Relevance 165

D.1 Retrieval Functions 166
D.2 Text Similarity Measures 168
D.3 Topic Models 170

1

INTRODUCTION

1.1 Motivation

1.2 Contributions

1.3 Outline

1.1 MOTIVATION

Data is increasingly becoming a valuable commodity, with personal and business decision making becoming more data-driven. As a result, tools that provide efficient data management, analysis or visualization functionality continue to gain in importance as they enable users to obtain information from their data. This trend is fostered by two concurrent conditions: a wider accessibility of fine-grained data on the one hand and technological progress in large-scale data processing on the other hand.

The Web takes on a special role in this data and information oriented society. Described as an *abundance of accessible information* (Kurland, 2006), the Web is an interlinked collection of documents which cover every topic conceived by people. We can find documents with detailed product information, articles imparting encyclopedic knowledge, presentations of state-of-the-art scientific results or reports on current financial data. The Web provides a platform for people to share all of their data as well as to satisfy their diverse information needs. With free access to the data, predominantly through highly efficient Web search engines, it is an important resource for users, including a significant amount of private individuals, but also professional users or organizations and enterprises, to get the answers they need. These information needs and application scenarios for Web data are plenty and diverse, ranging from consumers who wish to compare products or companies analyzing market trends to journalists who track government spending.

In addition to such user-centered applications, the Web also continues to grow in importance as a corpus for computational linguistics. Due to its sheer size and domain coverage, it provides a comprehensive account of the expressiveness and linguistic variations of natural language (Halevy et al., 2009). As a result, we can identify relations between words or phrases, such as synonymy or polysemy, by deriving statistics from the Web. Applications such as spell checking and machine translation, but also data integration, all benefit from the insights obtained from such a large corpus of language data (Kilgarriff et al., 2003). And beyond these applications, there is likely more potential in Web data for further use cases that have not yet been explored.

A special category of Web data, that has only come to the attention of researchers and application developers quite recently, is structured data stored in *tables*. Not natively supported by the text-centric approach of Web search engines, tabular data has received much less consideration compared to unstructured, textual data. Yet, the Web contains a considerable amount of tables with the same topical diversity. For example, Yakout et al. extracted about 154 million tables containing mostly relational data from Web pages (Yakout et al., 2012). Moreover, the growing Open Data trend, which sees public organizations and government bodies publish their data on designated platforms to increase transparency and accountability, further raises the amount of high-quality structured data that is freely available on the Web. Tables often contain facts about real-world entities, such as people, organizations or products, about their types as well as relationships between these entities. In other cases, they contain statistical data, for instance the results of scientific experiments.

Great potential lies in the content of these tables on the Web and a number of applications have emerged that benefit from this rich resource of structured data. An application that has received significant attention both in the research community as well as from commercial service providers is a specialized search engine for structured data (Balakrishnan et al., 2015), which provides native support for fact or entity search. Instead of returning relevant documents that require further handling from the user, direct answers to fact lookup queries can be returned, which is much more convenient

for users (Yin et al., 2011). Alternatively, the structured data found in Web tables can be used for the automated construction of large knowledge bases, such as Google's *Knowledge Vault* (Dong et al., 2014). In turn, these knowledge bases can then be used to support and improve a wide range of data analysis tasks. In fact, automating the construction of large-scale knowledge bases has recently been identified as one of the most important research directions by the database research community (Abadi et al., 2014). Beyond that, there are many more applications for tabular Web data, including situational data analysis or computational linguistics.

Compared to unstructured text, tables have several beneficial features. First of all, the structure of a table informs its semantics by reflecting logical relations between the data. Due to this regular structure, tables are generally less ambiguous than free text, which makes it easier to interpret the content algorithmically (Limaye et al., 2010). Additionally, tables provide a compact representation for multiple similar instances that have the same attributes, thus reducing the extraction and interpretation effort compared to freeform text. However, many traditional information extraction and retrieval techniques, including document-centric Web search, are not well suited for Web tables, as they generally do not consider the role of the table structure and layout in reflecting the semantics of the content. As a result, they cannot fully take advantage of this rich data source (Limaye et al., 2010). Therefore, in order to keep the content of Web tables from remaining underutilized, a designated Web table recovery and understanding process is required. The objective of this process is to recover and expose the entities and relations underlying the tables. For humans, understanding the information contained in Web tables is often an easy task, as the majority of these tables are intended for human consumption. However, to enable further utilization of the data and considering the scale of the Web, there is a great need for an automated algorithmic processing of the tables. Here, we face a wide range of challenges. First of all, the tables are *embedded* in the Web, which means they must be located and extracted (Balakrishnan et al., 2015). To recover the semantics of a table, we often need to consider the interplay between the structured data in the table and the unstructured data of the context (Cafarella et al., 2011a). Constraints of the table data are often only mentioned in the caption or surrounding text (Yin et al., 2011). Furthermore, the answers to queries can be distributed among multiple tables from different sources, raising a need for integration of Web tables (Halevy, 2004). However, no uniform schema or controlled vocabulary exists on the Web (Limaye et al., 2010). The tables are very heterogeneous, since most of them have been developed individually, leading to different choices in the design of the schema and the selection of attribute labels. And in addition to that, there is no centralized quality control in place, resulting in significant variation in the quality of the data as well as its description.

In recent years, a lot of research effort has been put into addressing these challenges in order to develop a process for Web table understanding. Significant contributions have been made regarding the identification of relational tables on the Web (Cafarella et al., 2008b), finding related tables (Das Sarma et al., 2012), or matching attributes in the tables to entries in a knowledge base, in order to understand the content (J. Wang et al., 2012). However, as pointed out by Yin et al. (2011), the precision and coverage of the data extracted from Web tables is often still quite limited. As Web table recovery and understanding is very complex, with many different challenges, many techniques developed so far make simplifying assumptions about the table layout or content to reduce the complexity. Thanks to these assumptions, many subtasks become manageable. However, they also imply a limited scope or a limited accuracy, if applied to tables that do not conform to these assumptions. In this thesis, our goal is to extend the Web table understanding process with techniques that enable some of these

assumptions to be relaxed, thus improving the scope and accuracy. We consider extensions in various aspects of table understanding, including context recovery and conceptualization.

1.2 CONTRIBUTIONS

In this thesis, we provide the following key contributions:

1. We provide a comprehensive analysis of tables available on the Web, including tables embedded in HTML pages as well as tables published on designated Open Data platforms. We highlight characteristic features of Web tables as well as identify unique challenges that arise from these characteristics in the table understanding process. Furthermore, we propose a classification scheme in order to categorize these tables based on the structure and general semantics of the table layouts. The analysis also includes a survey of key applications for Web tables in order to identify significant requirements for Web table understanding (Braunschweig et al. (2012) and Eberius et al. (2012)).

2. We survey recent literature on table recovery and understanding, in general, and Web table understanding, in particular, to assess the state of research in the field and identify limitations and open issues in previous work. While general table recovery research has been surveyed before, we are not aware of any previous surveys of research related to Web tables.

3. We introduce an alternative method to incorporate layout classification into the table understanding process. In the same context, we review, consolidate and extend features and classification schemes proposed in the literature and utilize feature selection to identify the most relevant features for the tasks involved.

4. We study the role of context information for Web table understanding and analyze the relevance of different context resources. To reduce the amount of irrelevant and misleading information in large context segments, we propose a selection algorithm that extracts topically coherent, relevant paragraphs. Reducing the context to relevant information, we improve the accuracy of various subsequent tasks that rely on information extracted from the context.

5. We propose an extraction approach for the recovery of attribute-specific information from the context of a table. Instead of using context as a general descriptor of the table content, these attribute-specific annotations provide a richer, more accurate description, from which table search or integration tasks can benefit. Our approach also serves as an extension to existing header recovery techniques based on external knowledge bases (Braunschweig et al. (2015a)).

6. Finally, we study Web tables at the conceptual level, specifying the task of *semantic normalization* to identify concept boundaries in complex tables. We derive a set of indicators to evaluate the relatedness of attributes in these tables and propose a normalization technique that identifies semantic concepts based on these indicators, in order to decompose the table accordingly. Our technique serves as a necessary preprocessing step for various Web table understanding tasks that assume a single semantic concept per table (Braunschweig et al. (2015b)).

1.3 OUTLINE

Figure 1.1 illustrates the outline for this thesis, with the remaining chapters organized as follows. In Chapter 2, we start with a review of the role and characteristics of tables in general, before outlining the foundations of the process involved in table understanding. We then shift our focus to Web tables in particular, in Chapter 3. We start with an analysis of the characteristic features of Web tables and present key applications for Web tables to highlight essential requirements that must be met by the table understanding process. Reviewing existing research in the field, we then derive a set of limitations and open issues of Web table understanding that we address in this thesis. In detail, we extend three subtasks of the table understanding process: (1) table layout classification, (2) context and header recovery, and (3) conceptualization, with each subtask covered in a separate chapter in the main part of this thesis. Chapter 4 focuses on incorporating layout classification into the understanding process. In Chapter 5, we study the role and importance of contextual information for the understanding and utilization of Web table content. In the first part of this chapter, we analyze the relevance of various available context resources with respect to the table content. In the second part, we then utilize the context to extract supplementary information that extends the description of attributes in the table. Chapter 6 addresses the conceptual model of Web tables, utilizing semantic normalization to expose the semantic concepts described in the tables. Finally, in Chapter 7, we conclude this thesis by summarizing our findings and discussing directions of future work.

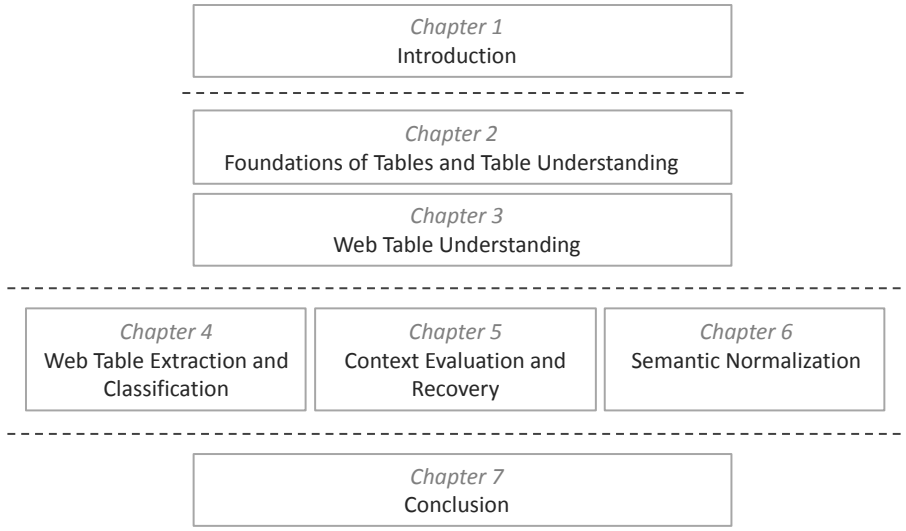


Figure 1.1: Organization of the chapters in this thesis.

2

FOUNDATIONS OF TABLES AND TABLE UNDERSTANDING

2.1 Tables and Relations

2.2 Automated Table Recovery

In this chapter, we introduce the key concepts and research areas that form and influence the subject matter of this thesis. First, we study the notion of a *table* and the different characteristics associated with this notion when used in different application scenarios. After that, we define *table understanding* in the larger context of *table recovery*, taking into consideration previous research in the field.

2.1 TABLES AND RELATIONS

Tables are frequently found in printed documents, such as books or journals, as well as digital documents, such as Web pages or presentation slides. But they also represent an important concept in relational databases and spreadsheets. Tables are a versatile tool for the representation and communication of relational or similarly structured data (Embley et al., 2006). Through a two-dimensional layout, they provide a compact visualization of the data that particularly facilitates the *search* and *comparison* of values of interest (Zanibbi et al., 2004). Depending on the domain or intended use, however, table layouts exhibit a structural diversity that makes it difficult to give a concise definition of what exactly constitutes a table (Xinxin Wang, 1996; Hurst, 2000).

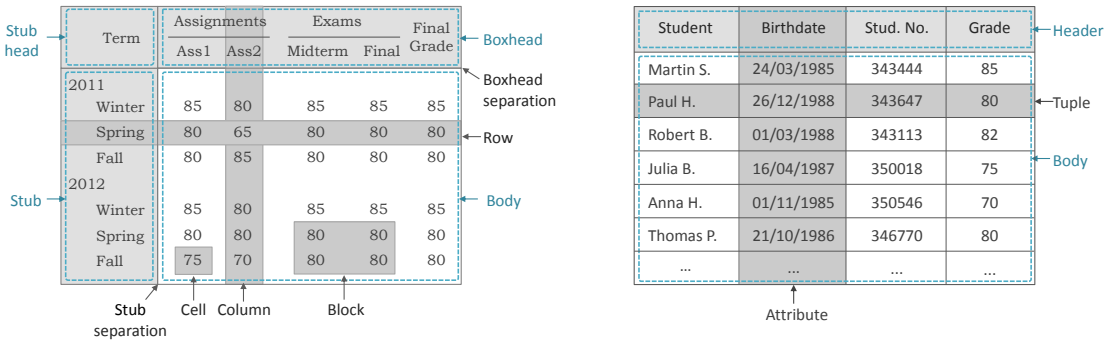
In general, tables organize the data in *cells* that are arranged in *columns* and *rows*. We differentiate between *data cells* and *label cells* (Pinto et al., 2003). Data cells form the *body* of a table and contain attribute values, predominantly words or numbers, but also more complex data objects, such as formulas, graphics or, in the case of nested tables, another table (Lopresti et al., 1999). These values represent the main content of a table, the target of a search or comparison. In contrast, the purpose of label cells is to characterize or describe the values in the body. They contain labels to *name* individual attributes in the tables, as well as additional categories or dimensions to *order* or *group* attribute values. Together, these label cells form an *indexing structure* for the values in the body (Zanibbi et al., 2004).

Within the two-dimensional structure of a table, the label and values of the same attribute are aligned along one axis in order to facilitate an efficient comparison. In most domains, a vertical alignment, with the label placed at the top of the column, is used. However, occasionally, a horizontal alignment is more convenient.

The meaning behind a table is conveyed through the layout (Pinto et al., 2003), with some of the information implicit in the table structure. In general, there is no universal way to read tables. Each table represents a particular view on the underlying data. Thus, for different purposes, the same data can be presented in different tables. To understand a table's meaning correctly, we often require additional background knowledge and contextual information.

Tables represent a spatial arrangement of content. Similar structures are also frequently used for layout and content composition. What sets a *genuine table* apart from more general grid-like arrangements is a reasoned structuring of the content that also reflects the logical relations between the data. A genuine table contains coherent, potentially redundant information. In short, the basic characteristics of a table can be summarized as follows:

Definition 2.1.1 (Table)**.** A table is a compact form of representation of structured data that facilitates the search and comparison of its content. In a two-dimensional arrangement of rows and columns, the data is organized to reflect its inherent logical relations. A designated indexing structure of labels and categories is employed to efficiently locate individual elements in the table.



(a) Document Table

(b) Relational Table

Figure 2.1: Contrasting the complex layout of document tables and the simple layout of relational database tables.

While Definition 2.1.1 holds for tables in general, in different application scenarios tables exhibit very distinct features. In the following, sections we will look at both, *document tables* and *database tables*, in more detail.

2.1.1 Tables in Documents

Tables in printed or digital documents are primarily intended for human consumption. The physical layout of these tables is often illustrated via graphical features, such as line-art and spacing, that are used to delimit individual cells (Hurst, 2000). As part of an often larger document, the principal purpose of these tables is to visualize specific relations between the data and communicate these characteristics to the reader of the document. Thus, the layout and content are generally fixed and not meant to be modified by the user.

In the literature, the *Wang notation* is commonly adopted to describe the compositional structure of document tables (Xinxin Wang, 1996). As illustrated in Figure 2.1(a), the notation recognizes four main sections in a table: the stub head, the stub, the boxhead, and the body. The stub covers the row headings to the left, while the boxhead covers the column headings. Together, these two sections contain labels, organized into categories, which form the indexing scheme for the entries in the body of the table. Not every table features all four sections, as especially the stub head, but also the stub, may be empty. However, it is common for document tables that the logical dimension of the table exceeds the two dimensions of the grid structure, with multiple categories being recorded in the stub or boxhead. In the table in Figure 2.1(a), for instance, the stub records the categories *year* and *term*. Such a case often leads to complex reading paths in order to locate entries in the body of the table (Hurst, 2000).

Structures that feature ordered and grouped values are prominent among document tables, as they often represent complex relations in the data. However, the layout is not always designed to only convey the semantics of the data. Sometimes, it simply caters to spatial restrictions in the document, without carrying specific meaning, which must be considered when reading a table.

Since the tables are generally published as part of a larger document, and not in isolation, it is also

common that some of the information required to understand the semantics of a table is not mentioned in the table directly, but in the surrounding context, which can be a caption, a title or accompanying text (Embley et al., 2006). Attribute labels, for instance, are often missing or vague, and only specified in the context. Also, categorical attributes, which apply to all values in the table, are commonly factored out to the context (Embley et al., 2005).
Despite the structural complexity and the often implicit semantics, humans are generally capable of understanding the meaning behind document tables, whereas it is very challenging to interpret these tables algorithmically. To keep the amount of data to an extend that is manageable for humans and due to the spatial restrictions of the medium, tables in documents are often small in size.

2.1.2 Tables in Relational Databases

The table is the primary *database object* to hold relational data in a database and, thus, represents a fundamental concept in relational database management systems (RDBMS). It acts as a *logical data structure*, meaning that the tabular organization of the data is merely *conceptual* and independent of the actual physical storage that is designed to optimize memory utilization and query performance. In database terminology, the terms *table* and *relation* are often used interchangeably. However, in this thesis we differentiate between these two terms.
The term *relation*, in the data modeling sense, was first introduced by E.F. Codd as the central element of the *relational database model* (E. F. C. F. Codd, 1970) to describe the logical structure of data. Based on its set-theoretic notion, a relation is defined over a set of *domains*, each representing an *attribute* of the relation. A domain, alternatively called a *dimension*, is simply a set of all possible values associated with this domain. The relation itself is then a subset of the Cartesian product of the attribute domains. The members of this subset are called *tuples*. A tuple is a sequence of not necessarily unique elements, with one value from each attribute domain. The association between tuple elements and attributes is established by either maintaining an ordering of elements within each tuple, or by mapping attribute labels to values (Embley et al., 2006). In summary, a relation can be defined as follows:

Definition 2.1.2 (Relation)**.** An n-ary relation R is a set of *tuples*. Each tuple is a sequence of n values V_i, with $V_i \in D_i$ for $1 \leq i \leq n$. Each D_i is a set of values that constitutes the ith *domain* of R. The domains of a relation are not required to be distinct.

In contrast to a relation, a table, in the context of an RDBMS, is a data structure that is utilized in order to organize, process and visualize relational data, and is not itself part of the relational model (E. F. C. F. Codd, 1970). A database table features a layout consisting of a *header*, which holds the attribute labels, and a *body*, which holds the tuples of a relation. When displayed, the header is positioned at the top (see Figure 2.1(b)). This simple layout with clear reading paths enables database tables to be processed automatically, as they are mainly designed for algorithmic consumption of large datasets. The layout generally features no nested headers or a stub. Instead, *primary keys* defined over the relation extend the indexing scheme to address individual tuples in the body. To reference the data, relations as well as attributes must be named. In a relational database, tables are typically *normalized* to reflect application semantics and avoid data inconsistencies. The normalized base tables can be linked explicitly via *foreign keys*, forming larger database schemas. Both, the table structure and the content, can be modified using data manipulation operations, including *INSERT*, *UPDATE* and *DELETE*.

In an RDBMS, the result set of a query is also represented as a table, even when only a single value is returned. Similarly, the result of frequently performed queries that require expensive join and aggregation operations may be recorded for higher efficiency in so-called *materialized views*, which also have the form of a table. As a result, these tables often combine data from multiple tables (Hurst, 2000).
As discussed previously for document tables, the two-dimensional layout of tables does not lend itself naturally to multi-dimensional data, such as time-varying data, scientific data or sensor data (Shoshani et al., 1985). Similarly, it is difficult to reflect classification hierarchies for categorical attributes in the simple layout of relational database tables (Rafanelli et al., 1990). To enable efficient analysis of such multi-dimensional data at fine granularity, alternative data models are often required.
In the context of *online analytical processing* (OLAP), the multi-dimensional data model splits data into separate fact and dimension tables for efficient analysis and reporting. While fact tables store the values of measures or quantitative attributes, the dimension tables reflect the categories and category hierarchies used to index or describe the facts. The *OLAP cube* provides a logical abstraction on top of these tables for a more coherent access to and analysis of multi-dimensional data (Gray et al., 1997).
In contrast, *scientific databases management systems*, which target the analysis of large volumes of high-dimensional, aggregate data, have seen a trend away from the relational model to more complex data models (Rafanelli et al., 1990). Flexible, array-like data structures are frequently used to provide natural support for multi-dimensional data, with additional data structures to reflect the relationships between dimensions and hierarchies within categories. A more recent example is the *Array Data Model* employed in *SciDB* (Stonebraker et al., 2013).
Overall, tables suitable for algorithmic access generally have simple layouts, with additional data structures required to support complex, multi-dimensional data.

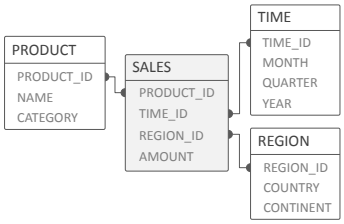
2.1.3 Linking Document and Database Tables

In summary, tables in printed or digital documents and tables in RDBMS are two inherently different concepts. They share the same basic layout that provides a compact representation by arranging data in rows and columns, and both feature a designated indexing scheme of labels to reference values in the table.
However, their respective purposes and application scenarios differ significantly, which is reflected in some of the tables' features. Database tables are designed to hold large amounts of data and facilitate efficient *automated* processing and querying. Therefore, they store the data in its original format in simple, modifiable data structures. In contrast, document tables aim to present a detailed view or highlight certain characteristics, often in the form of summaries or aggregates, of the underlying data to human readers. A more complex layout provides the expressiveness required to effectively communicate this information. Consequently, document tables can represent a *view* over multiple (linked) database tables (Hurst, 2000), as shown in Figure 2.2. Unlike (materialized) views in RDBMS, they are not limited to the simple layout of database tables.
The semantics of the data is also recorded differently in these two types of tables. To enable automated processing through applications, database tables reflect their semantics explicitly via unique relation and attribute labels as well as referential and integrity constraints. Document tables, on the other hand, often only communicate their semantics implicitly through the table layout or via associated context information. While this is relatively easy for humans to comprehend, it is much more chal-

Average Sales (in Mio. $)

Region	2014		2015	
	Q1	Q2	Q1	Q2
Europe				
UK	14	15	13	13
France	21	20	24	23
Asia				
Japan	10	10	11	10
China	63	64	66	67

(a) Documents Table



(b) Database Schema

Figure 2.2: Representation of multidimensional data.

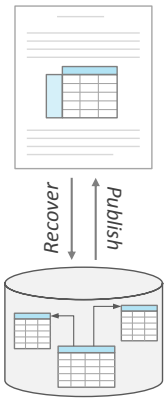


Figure 2.3: Transformations.

lenging to process algorithmically.

Although document and database tables are often created separately and under very different circumstances, there is a need and application for transformations between the two formats (see Figure 2.3). On the one hand, views on database tables may be published for reporting and data sharing. Publishing the data potentially involves loss of schema information, a de-normalization of tables as well as the anonymization of sensitive personal data. However, guided by the known semantics of the table content, systemizing this process is relatively easy to realize.

On the other hand, public tables in documents may be recovered and imported into a database to facilitate reuse, integration with other datasets, as well as automated processing and analysis of the data. The main challenges involved in this process include maintaining the table context, recovering implicit semantics as well as resolving the complex table structure. Without knowledge of the intention with which the table was published, recovering and especially understanding document tables is very difficult. Considering the transformations in both directions, recovering the data stored in document tables for reuse in a database is considerably more complex and, thus, harder to accomplish algorithmically.

The focus of this thesis is the recovery of document tables. Thus, we take a closer look at the process involved and review relevant literature in the subsequent section.

2.2 AUTOMATED TABLE RECOVERY

Table recovery describes the process of retrieving structured data presented in tables in printed or digital documents. In the literature, this process is also sometimes referred to as *table processing* (Lopresti et al., 1999, 2000). To avoid confusion with other tasks in the wider domain of processing structured data, we will use the former term in this thesis. The main objective of table recovery is to enable the *reuse* of the data in applications, independent of the original data format or medium (Embley et al., 2006). This includes the ability to query and mine the data in the same way as database tables.

2.2.1 Process Overview

The complete table recovery process, depicted in Figure 2.4, can be divided into four consecutive steps (Göbel et al., 2012). The necessary tasks performed in each of these steps depend on the source format and medium of the original table as well as on the target application. If the source document is stored in an image file, for instance, these tasks include image analysis and optical character recognition to retrieve machine-readable text, whereas text files, and, more generally, document formats already provide the textual content in a machine-readable format. Note that this process requires the documents to exist in a digital format, which means that documents on paper must be scanned first.

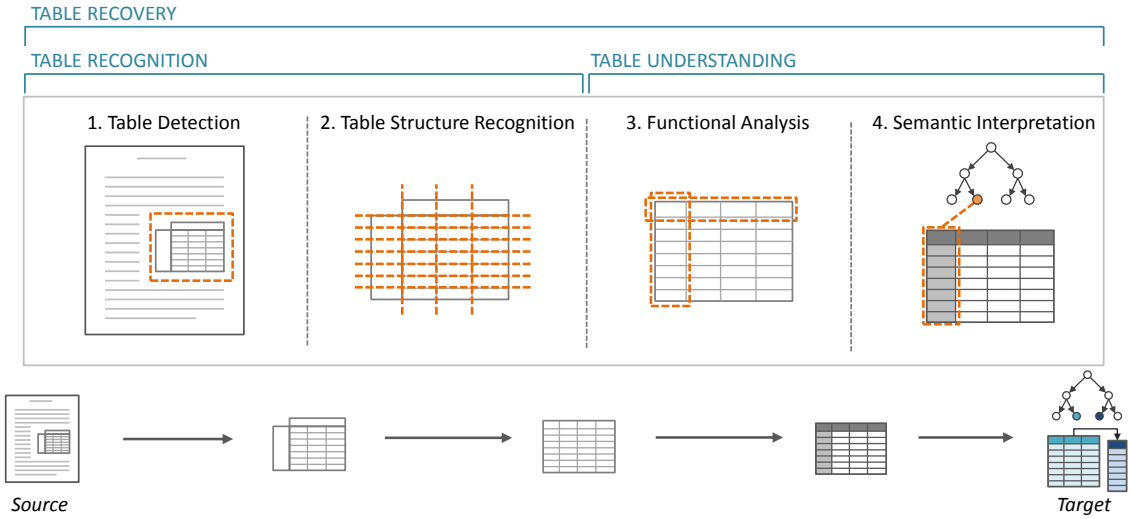


Figure 2.4: Overview of sub-processes involved in table recovery.

1. **Table Detection:** The purpose of this first subprocess is locating the table within the source document. This involves identifying the outer boundaries of the table.

2. **Table Structure Recognition:** This second subprocess aims to reconstruct the cellular structure of the table. This requires the identification of cell boundaries as well as the recovery of cell alignments. It also includes the reconstruction of hierarchies for merged and nested elements. The output of this step is a description of the *physical structure* of the table.

3. **Functional Analysis:** In this subprocess, the meaning of the table structure is analyzed. The function or role of each cell in the table is recovered, separating label cells from data cells. After that, each data cell is associated with its corresponding label cells. The result is a description of the table's *logical structure*.

4. **Semantic Interpretation:** In the final subprocess, the content of the table is analyzed, in order to identify the entities, attributes and relationships presented in the table. This includes analyzing the context of the table, such as the title, caption or surrounding text.

In the literature, the subprocesses of table detection and structure recognition are sometimes combined and collectively referred to as *table recognition* (Zanibbi et al., 2004), while the combination of functional analysis and semantic interpretation is referred to as *table understanding* (Embley et al., 2006).

2.2.2 Survey of Related Work

Aspects related to table recovery have been studied from many different angles in various communities, including image analysis, information retrieval, question answering, and data mining. In the scope of this research, tables in various types of documents have been explored, ranging from images of scanned documents (Cesarini et al., 2002), ASCII text (Ng et al., 1999; Pyreddy et al., 1997; Pinto et al., 2003; Hurst, 2003) and PDF documents (Göbel et al., 2012) to HTML files (Pinto et al., 2002; Embley et al., 2005; Y. Wang et al., 2002) and published spreadsheets (Z. Chen et al., 2013, 2014; Adelfio et al., 2013).
Research initially focused on table recognition, especially the detection of tables in scanned documents and free text. It was later extended to not only address the physical, but also the logical structure of the tables, by differentiating between label and data cells. It has been shown that the detection and structural analysis are reasonably well suited to be carried out algorithmically, yielding high quality results for a large number of table layouts. More recently, research has moved its focus to the semantic interpretation of tables, which is significantly harder to automate, as it requires a more comprehensive analysis of not only the table itself, but also its context. Tables 2.1 and 2.2 provide an overview of previous work related to table recognition and table understanding, respectively. Although assigned to individual subprocesses in the tables, many of the proposed approaches actually contribute to several of the subprocesses in the table recovery pipeline. While some techniques are based on domain-specific models (for instance, the work of Embley et al. (1999, 2005)), others follow a general purpose approach independent of specific domain knowledge (such as the work of Pyreddy et al. (1997)).

Table Recognition
Table detection and *structure recognition* are often performed simultaneously. In general, we distinguish between *rule-based* approaches and *learning-based* approaches (especially classification). Popular classifiers include decision trees (Ng et al., 1999) and conditional random fields (CRF) (Pinto et al., 2003). Apart from locating tabular structures in documents, table detection is also concerned with the identification of *genuine* tables, as opposed to tabular arrangements utilized for layout purposes. Structure recognition is especially relevant for tables in scanned images and free text. Different tech-

Table 2.1: Classification of previous research studying table recognition.

Table Detection	Structure Recognition
• Rule-based table detection (Pyreddy et al., 1997)	• Row classification (Pinto et al., 2003)
• Learning-based table detection (Pinto et al., 2003)	• Rule-based structure recognition (Pinto et al., 2002)
• Genuine table identification (Cafarella et al., 2008b)	• Constraint-based structure recognition (Hurst, 2003)

niques consider table components at different composition levels. Some approaches only identify row boundaries, while more advanced techniques also identify column boundaries, leading to individual cells being recognized. Usually, rules or constraints are defined to recover column boundaries from the spatial alignment of characters. In contrast, some representation formats, such as HTML and spreadsheet files, provide natural support for cellular structures, which renders structure recognition techniques unnecessary.

Table Understanding
The *functional analysis* is strongly affected by the complexity of the table layout in general and the complexity of the indexing scheme in particular. While simple relational layouts can be analyzed by means of learning-based techniques, more complex layouts often require a more specialized, rule-based inference to achieve good quality results. Seth et al. (2010) rely on context-free grammars to retrieve column header hierarchies from tables with complex indexing schemes. In contrast, Z. Chen et al. (2014) use extensive probabilistic graphical models to retrieve the most likely label hierarchy. The same models are also used to map the retrieved labels to data cells in the tables. In contrast to table detection and structural analysis, which involve a small set of clearly defined tasks, the components of *semantic interpretation* of tables are still evolving and often more vague. Previous research includes the inference of headers for tables with empty attribute headers, the discovery of entity mentions, the establishment of attribute constraints, as well as the recovery of context information. The final goal, and often the main focus of semantic interpretation, is the *conceptualization* of tables, which involves establishing the semantic concepts in a table by mapping its attributes to entries in a reference knowledge base (J. Wang et al., 2012). This facilitates the reuse of the data outside its original context.

2.2.3 Application to Tables on the Web

The attention directed at table recovery research has increased considerably in recent years, largely due to general advances in large scale data processing on the Web. These advances not only enabled the efficient extraction of large amounts of data, but also the modeling and population of extensive conceptual models of the world, such as taxonomies and general purpose knowledge bases (Wu et al., 2012). The utilization of such conceptual models ultimately facilitates the domain-independent conceptualization of tables. The development of sample applications further showcases the potential

Table 2.2: Classification of previous research in table understanding.

Functional Analysis	Semantic Interpretation
• Rule-based header detection (Pinto et al., 2002) • Learning-based header detection (Cafarella et al., 2008b) • Resolving hierarchical column headers (Seth et al., 2010) • Probabilistic label-to-data mapping (Z. Chen et al., 2014)	• Header inference via reference matching (Cafarella et al., 2008b) • Entity discovery (Quercini et al., 2013) • Subject column detection (Venetis et al., 2011) • Column unit annotation (Sarawagi et al., 2014) • Recovering attributes from context (Cafarella et al., 2009) • Conceptualization (J. Wang et al., 2012)

of data embedded in tables and, thus, leads to increased interest in advancing the table recovery process.
In this thesis, we also address the recovery of Web tables in particular, especially the understanding of Web tables. Therefore,in the subsequent chapter, we take a closer look at the different types of tables encountered on the Web and the implications that result from these characteristics for the process of Web table understanding.

3

WEB TABLE UNDERSTANDING

3.1 Tables on the Web

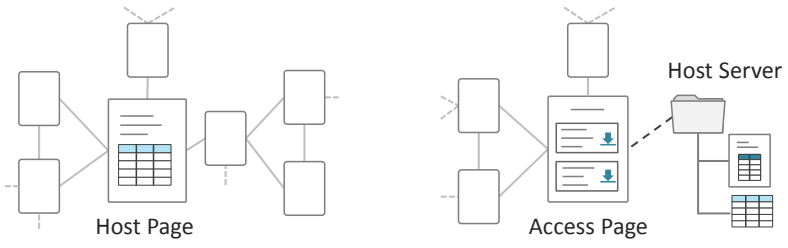
3.2 Web Table Applications

3.3 Revisiting Table Recovery

As established in the previous analysis of the table recovery process, the individual processing steps necessary to recover data from document tables are determined by the medium and format of the source documents as well as specific application requirements. As our first contribution, we provide a detailed study of Web tables and their applications in order to compile the process required for Web table recovery. First, we take a look at the diverse set of tables available on the Web (Section 3.1). After that, we survey potential applications for Web table data to derive necessary requirements (Section 3.2). We then revisit the table recovery process, with the specific characteristics of Web tables in mind, and examine previous research in the field (Section 3.3). Finally, we highlight the open challenges in Web table understanding and outline how these challenges are addressed in the remainder of this thesis (Section 3.3.4).

3.1 TABLES ON THE WEB

The Web is a valuable, freely accessible resource of data covering every imaginable topic, a significant part of which is stored in tabular form (Cafarella et al., 2011b). We distinguish two kinds of sources for tables on the Web, as illustrated in Figure 3.1. Note that we only consider *genuine* tables, which present meaningful relations between data. Tabular structures used for layout purposes, which are also very common on the Web, are not taken into account.
First, tables can be directly *embedded* in Web pages, as HTML tables using designated markup (Gatterbauer et al., 2007) or as plain text tables, using line-art and spacing for alignment (Pinto et al., 2003). These tables are *displayed* with the rest of the page content for online consumption, which imposes limits on the size of the tables. The *host page* provides the context for the table, in the form of titles, captions or surrounding text. Alternatively, tables can reside in *external documents* that are *linked* to a Web page, which provides access to these documents as well as additional metadata. The documents can be found in a number of open and proprietary formats and are often published as part of a larger collection in designated data portals, such as *Open Data* platforms.
In the literature, the term *Web table* is often used solely for tables embedded in HTML (see, for instance, the work of Yin et al. (2011)). In this thesis, we adopt a less restrictive meaning that also includes other formats. These tables often share many similarities with HTML tables regarding layout, accessibility and domain coverage and, therefore, require similar processing steps.



(a) Embedded Table (b) External Table

Figure 3.1: Sources for genuine tables on the Web.

```
| Country  | Capital | Population |
+- - - - - +- - - - -+- - - - - - - +
| Germany  | Berlin  | 81 million |
| France   | Paris   | 66 million |
```

(a) Plain text

```
Country, Capital, Population
Germany, Berlin, 81 million
France, Paris, 66 million
```

(b) CSV

```
<TABLE>
   <TR>
      <TH>Country</TH>
      <TH>Capital</TH>
      <TH>Population</TH>
   </TR>
   <TR>
      <TD>Germany</TD>
      <TD>Berlin</TD>
      <TD>81  million</TD>
   </TR>
   <TR> … </TR>
</TABLE>
```

(c) HTML

Figure 3.2: Representations of a sample table in various formats.

3.1.1 File Formats

On the Web, we can encounter tables in many different file formats, which, in addition to HTML, include plain text formats (e.g. ASCII), image formats (e.g. JPEG), general document formats (e.g. PDF), as well as specialized exchange and storage formats for structured data (e.g. CSV, XML) (Braunschweig et al., 2012). The complexity of algorithms required to recover the physical and logical structures of tables strongly depends on the respective file format and the functionality provided by the format to encode this structure. Figure 3.2 shows three examples of how the same table can be encoded in different formats.

File formats such as plain text and image formats provide no designated tags or markers to encode the tabular structure. Instead, line-art, whitespace and text formatting are employed to represent the structure of tables (Embley et al., 2006; Hurst, 2000).

Formats that provide special functionality to encode only the physical, but not the logical, structure of the tables include exchange formats such as CSV, where tuples are separated by line breaks and tuple entries are separated by a designated delimiter. There is no distinction between label and data entries. CSV files require a regular cell structure, as there is no native support for nested cells or metadata (Repici, 2015).

More functionality is provided by formats that employ a *markup language*, such as SGML, HTML or XML. In addition to encoding the physical structure of tables, designated markup is used to encode (some of) the tables' logical structure (Zanibbi et al., 2004; Lopresti et al., 1999). These *tags* are mostly used to mark tables in larger documents and to identify label and data cells within the table.

The more structural information about a table is already recorded in the document, the less work is required to extract this information algorithmically. While the use of markup languages simplifies the table extraction process, there are new challenges to consider during table recovery. On the Web, markup and delimiters are often misused or exploited for other purposes (Embley et al., 2006; Zanibbi et al., 2004). For instance, HTML `<TABLE>` tags are more often used for page layout purposes than to display actual data tables (Cafarella et al., 2008b).

We limit the file formats considered in the remainder of this thesis to those that encode at least the physical structure of the tables, to reduce the amount of preprocessing. This includes tables that are either embedded in HTML or stored externally in CSV or XML documents. However, the techniques presented here may also be applied to all other tables, once the physical structure has been extracted.

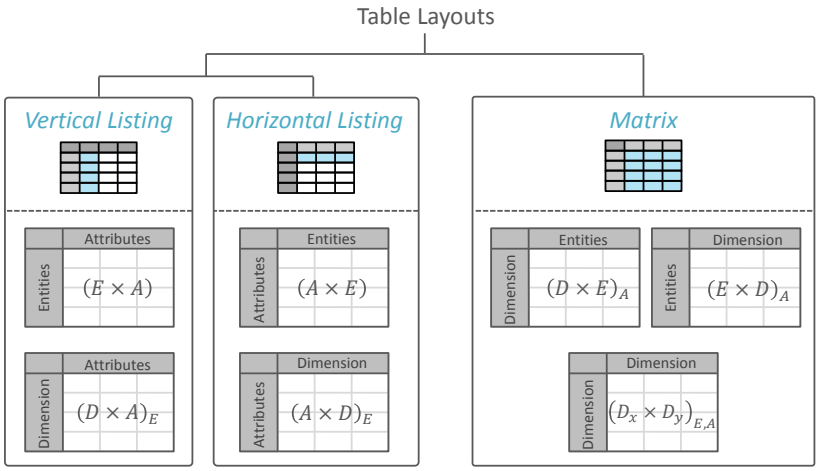


Figure 3.3: Categorization of Web tables, based on layout characteristics.

3.1.2 Table Layout

Individual layouts of tables on the Web vary greatly with respect to the position and orientation of headers and the alignment of tuples. While tables exported from a DBMS usually have a very simple, relational structure, tables intended for human consumption, as those embedded in Web pages, can be much more complex and diverse (Embley et al., 2006; Hurst, 2000).

In the literature, different classification schemes have been proposed to categorize table layouts, with the most extensive taxonomies compiled by Crestan et al. (2011) and Lautert et al. (2013). Both taxonomies consider different types of genuine tables carrying relational data as well as tables used for layout purposes. Crestan et al. also consider *enumerations* and *forms* in their taxonomy, which we do not regard as genuine tables, based on Definition 2.1.1. In addition to the main taxonomy, Lautert et al. also provide a secondary classification based on artifacts encountered in Web tables that are not supported by the relational data model, such as nested, merged or multi-valued cells. However, this second taxonomy is more a classification of features than of tables, which may contain multiple or none of the proposed artifacts.

Taking the classification schemes in the literature into account, we propose a general taxonomy for genuine Web tables, which is illustrated in Figure 3.3. In contrast to previous work, we do not include layout tables. The main classification contains three categories, based on the *alignment* of attribute values in the table. Values of the *same attribute* are either aligned in a *vertical listing*, a *horizontal listing* or forming a *matrix* (Crestan et al., 2011). Tables in the first two categories often list multiple attributes $A_1 \ldots A_n$. Vertical listings include relational tables, like the example in Figure 3.4(a), where the names of lakes as well as their respective areas are listed vertically. Horizontal listings are simply *transposed* variants of vertical listings. Depending on what aspect of the table data should be highlighted, either a vertical or a horizontal arrangement may be more suitable. In contrast, matrix tables only contain values of a single attribute or attribute type in the body. For instance, a table stating the population of various cities over a period of several years forms a matrix table, as all data cells contain population values. The transposed variants of matrix tables also form matrix tables.

	Lake	Area
1	Windermere	5.69 sq mi (14.7 km^2)
2	Kielder Reservoir	3.86 sq mi (10.0 km^2)
3	Ullswater	3.44 sq mi (8.9 km^2)
4	Bassenthwaite Lake	2.06 sq mi (5.3 km^2)
5	Derwent Water	2.06 sq mi (5.3 km^2)

(a) Relational Table [1]

Government[3]	
• **Type**	Mayor–Council
• **Body**	New York City Council
• **Mayor**	Bill de Blasio (D)
Area[2]	
• **Total**	468.9 sq mi (1,214 km^2)
• **Land**	304.8 sq mi (789 km^2)
• **Water**	164.1 sq mi (425 km^2)
• **Metro**	13,318 sq mi (34,490 km^2)
Elevation[4]	33 ft (10 m)

(b) Attribute-Value Table [2]

	Right-handed	Left-handed	Total
Males	43	9	52
Females	44	4	48
Totals	87	13	100

(c) Frequency Table [3]

Figure 3.4: Examples of various table layouts. All tables are extracted from the English Wikipedia.

These three categories provide a very general classification of Web tables, with a lot of variation within each category. One approach to further refine the broad taxonomy is to consider the logical dimensions of the tables. A relation underlying a table can be multi-dimensional, describing multiple similar *entities* E through a potentially large set of *attributes* A, each of which may have several *dimensions* D, such as temporal or spatial aspects. However, the two-dimensional (physical) table structure imposes a restriction on the dimensionality, so that often only a *subset* of the underlying dataset can be displayed in a single table. The significant attribute values (i.e. the ones that are to be searched and compared) are placed in the body of the table, while the remaining dimensions form the indexing scheme (i.e. the categories in the column and row headers (Zanibbi et al., 2004)). If more than two logical dimensions need to be displayed, attributes may be nested or grouped to factor in an additional dimension. An example for such a complex indexing scheme is shown in Appendix A. In the classification in Figure 3.3, however, we focus on tables with only two logical dimensions, as these cover the majority of tables on the Web. An example table for each variant is provided in Appendix A.

The first type of tables encountered in the category *vertical listing*, are tables that describe a set of attributes for a set of similar entities, with additional dimensions, most commonly temporal dimensions, set to a *fixed value*. These tables, denoted as $(E \times A)$ in the figure, are often simply called *relational tables* (Cafarella et al., 2008a) or *entity-attribute tables* (Yakout et al., 2012) and are among the most common table layouts on the Web. Here, the attribute labels are listed horizontally and entities (usually in the form of a *key attribute*, such as a name or identifier) are listed vertically. A respective transposed variant, with attribute labels listed vertically and entities listed horizontally, also exists in the category *horizontal listing*, denoted as $(A \times E)$.

The second type, denoted as $(D \times A)_E$, involves tables that describe a set of attributes for a *single entity*, using the second axis to factor in another dimension. The name of the entity in question is generally mentioned only in the context of the table. Tables in this category often present the evolution of certain attributes over time. Again, two variants exist. As a horizontal listing, this type of table layout is denoted as $(A \times D)_E$, with inverse labeling. A special case of this type of table, the so-called *attribute-value tables*, have a significant frequency on the Web. These tables vertically list a set of attributes for a single entity *without* additional dimensions, thus resembling a fact sheet. An example is depicted in Figure 3.4(b). Again, the name of the respective entity is often only referenced in the context, similar to other fixed dimension values.

[1] Source: `https://en.wikipedia.org/wiki/List_of_lakes_of_England`
[2] Source: `https://en.wikipedia.org/wiki/New_York_City`
[3] Source: `https://en.wikipedia.org/wiki/Contingency_table`

For *matrix* tables, we can also distinguish two different types. The first type, denoted as $(D \times E)_A$ or $(E \times D)_A$, if transposed, considers tables that compare a *single attribute* across multiple similar entities. The second axis is used for an additional dimension, most commonly a temporal dimension. The second type, denoted as $(D_x \times D_y)_{E,A}$ in the figure, describe a single attribute of *one entity* (or entity group) in two dimensions, which can be temporal, spatial or categorical. This type of indexing scheme is the predominant layout for results of statistical evaluations or surveys, in which case dimensions are often called *variables*. Common examples are contingency or frequency tables, such as the example in Figure 3.4(c).
Overall, the diversity of table layouts is one of the key challenges for Web table applications (Venetis et al., 2011).

3.1.3 Data Quality and Attribute Semantics

With a multitude of people contributing content and no universal quality control mechanisms in place, the quality of tabular data across the Web is very inconsistent. This includes the quality of the indexing scheme, i.e. the attribute labels, as well as the usability and accuracy of the table content. The quality of the indexing scheme is especially of interest in the table recovery process, as the completeness and the descriptiveness of attribute labels directly impact semantic interpretation efforts (Pimplikar et al., 2012).
Completeness of attribute labels is important for the correct identification of the relation underlying the data, including potential constraints. However, many tables on the Web have missing labels, so-called *implicit labels* (Embley et al., 2006), for some attributes, while others are missing the entire header (J. Wang et al., 2012). In Figure 3.5, we can see that in the Dresden Web Table Corpus (DWTC), for instance, more than 100 million attributes have no label (see Appendix B for more information on the corpus). Often, tables also feature *hidden* attributes. These are attributes where the *same value* holds for *every tuple* in the table (Cafarella et al., 2009; Ling et al., 2013). Instead of maintaining a column with a constant value, the value is factored out to the context. Often, these implicit attributes represent temporal or spatial selection constraints and are essential to infer the validity of the data (Venetis et al., 2011).
The descriptiveness of attribute labels refers to how well the meaning behind an attribute (and, eventually, the complete table) can be inferred from the label text, given such a label exists. In Web tables, we encounter various label characteristics that complicate the inference. In some cases, the labels are very generic, so-called *non-informative* labels, such as “name” or “value”. These type of labels provide little to no information on the meaning of an attribute. Figure 3.5 shows that such labels are among the most frequent on the Web. In other cases, labels contain long descriptions of the attribute, sometimes spanning multiple rows (Venetis et al., 2011). While these types of labels are easily understood by humans, complex text analysis is required to interpret them algorithmically. Finally, attribute labels can be abbreviated, which also requires specialized processing to recover the extended forms (Sorrentino et al., 2009). In addition to these challenges, we are also faced with the general linguistic ambiguity of words or phrases in the tables. Words can have different meanings depending on the context they are used in, just as different words may be used to describe the same concept or attribute (i.e. synonyms). Web tables do not adhere to a controlled vocabulary (Venetis et al., 2011) and, thus, feature a considerable linguistic diversity.

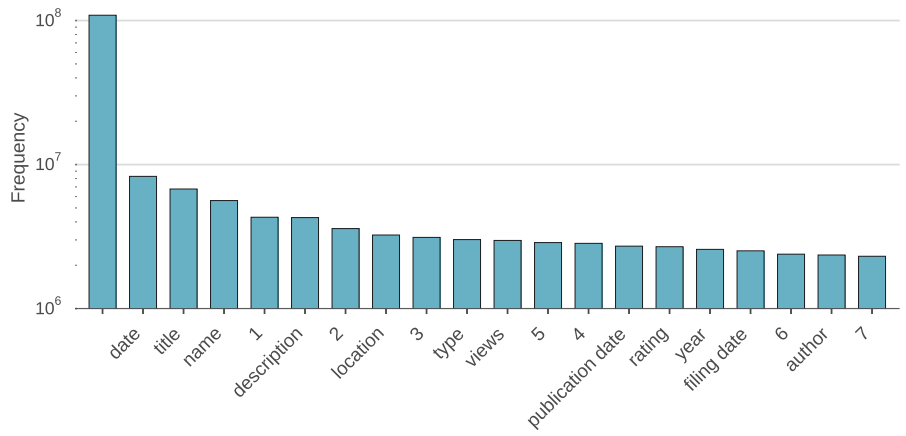


Figure 3.5: Frequency statistics of the 20 most frequently occurring attribute labels in the Dresden Web Table Corpus, including an empty label.

Semantically, tables on the Web often do not form *self-contained units*. Without consulting information in the context of the table (such as metadata, headlines, surrounding text or captions), it is very difficult, and sometimes impossible, to identify the meaning behind the data (Embley et al., 2006; Hurst, 2000), even if descriptive labels exist. For *attribute-value tables*, for instance, the corresponding entity name must be extracted from the context (Yin et al., 2011). Also, for tables with generic labels, the context often gives a more specific description of the content. Similarly, unit specifications, required to correctly interpret numerical data, are also sometimes removed from the column labels and stored in the title or caption (Sarawagi et al., 2014). An interpretation of the table content without considering contextual information will clearly miss important details.

3.1.4 Table Size

It can be observed that the size of a table is to some extent linked to its intended use. As previously highlighted in Section 2.1.3, document tables intended for human consumption are generally much smaller than tables intended for massive data consumption in a database management system. Additionally, Embley et al. (2006) point out that tables in documents intended for Web display, such as Web pages, tend to be smaller in size compared to tables in documents that are to be printed on paper. Due to the size restrictions, displaying wide tables on the Web is challenging, as it requires pagination while still sustaining the meaning of the table (Xinxin Wang, 1996). Therefore, especially tables embedded in HTML tend to be very small, both in column and row size.
This characteristic is highlighted in Figure 3.6, where the distribution of column and row sizes among the tables in the Dresden Web Table Corpus. The corpus contains millions of HTML tables with varying layouts extracted from Web pages, with an average column and row size of 4.16 and 12.70, respectively. Similar statistics for HTML tables have been reported by Yakout et al. (2012) and Cafarella et al. (2008b). In Figures 3.6, we can see that a large proportion of tables feature less than 10 columns and 20 rows. As a result, many tables provide only little content that can be utilized to infer the semantics or match the tables to user queries or other tables. Often, the small number of samples

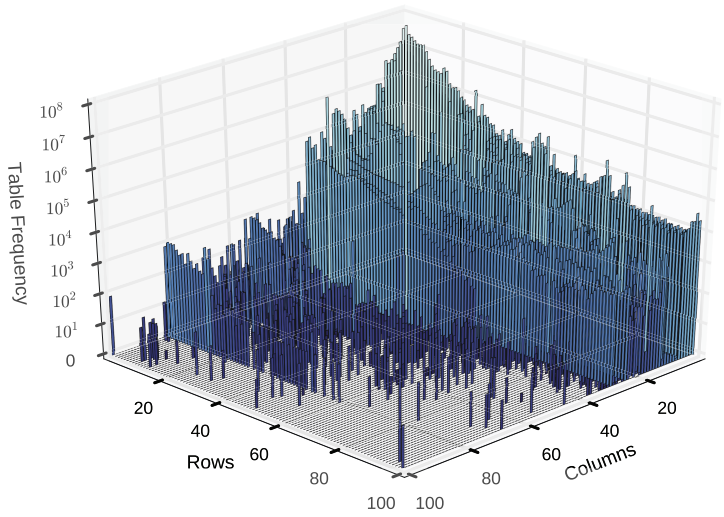


Figure 3.6: Distribution of table sizes with less than 100 rows and columns in the Dresden Web Table Corpus.

for each attribute do not sufficiently represent the statistics of the attribute domains. This leads to unique challenges for many table analysis tasks that rely on attribute statistics.
Although most tables on the Web are significantly smaller than most tables in enterprise databases, a long tail exists. In the DWTC, the largest tables have up to 113682 columns and 46743 rows. However, such large tables are very rare.

3.1.5 Corpus Size and Domain Coverage

Even though not every Web page contains tabular data, the Web still provides access to a substantial amount of tables, both embedded and as external documents. Previous efforts to extract genuine HTML tables report corpus sizes of 154 million (Yakout et al., 2012) and 147 million (Web Data Common - Web Tables[1]), respectively. The Dresden Web Table Corpus provides a similar scale, with roughly 145 million tables extracted from the Web (see Table 3.1). In addition to tables from Web pages, there is also a vast collection of tables published in linked documents. This collection is constantly growing due to ongoing *Open Data* and *Open Government* initiatives to increase transparency and accountability by publishing datasets collected by businesses and organizations in the public sector. Table 3.2 provides statistics for a selection of popular Open Data platforms. In these portals, data is published in so-called *datasets*, usually one or more data files which describe the same scenario and, thus, share the same metadata. The number of individual files is generally much higher. The statistics provided in the table are not limited to tabular data, but include other files as well. However, for each of these platforms, the majority of files represent tables or spreadsheets.
Similar to the Web itself, the tables accessible on the Web cover a wide range of topics from various domains. Some of these topics are economics, finance, transport, government expenditure, energy,

[1] `http://webdatacommons.org/webtables/`

Table 3.1: Size statistics for Web table corpora extracted from HTML.

Corpus	# of Tables
• WebTables (Cafarella et al., 2008b)	154 M
• InfoGather (Yakout et al., 2012)	573 M
• Web Data Commons - WebTables (`http://webdatacommons.org/webtables/`)	147 M
• Dresden Web Table Corpus (see Appendix B)	145 M

Table 3.2: Size statistics for a selection of Open Data platforms (*last checked 28.05.2015*).

Platform	# of Datasets
• data.gov.uk	24804
• data.gov	132088
• open-data.europa.eu	8501
• opendata.socrata.com	22849
• datahub.io	9582

environment, international relations or health. While this variety of domains turns Web tables into a rich data source, it also introduces new challenges. Domain-specific recovery techniques are not sufficient to examine a Web table corpus as a whole. Instead, open-domain approaches are required. However, the more domains are covered by the corpus, the more heterogeneity and ambiguity are introduced due to domain-specific terminology.

Summary

For an effective table understanding, as part of the automatic table recovery process, it is important to adapt individual processing steps to the characteristics of the data. While most of these characteristics complicate the recovery process, some characteristics can also be beneficial to the task. In summary, the main characteristics of tables on the Web are:

- **Heterogeneous Formats:** Tables exist in many different file formats, especially on Open Data platforms, with varying degrees of functionality to encode the physical and logical structure of a table. The less structural information is encoded in the file, the harder it is to recover the table structure algorithmically.

- **Diverse Layouts:** The Web features a great variety of table layouts. Indexing schemes vary in semantics, orientation as well as complexity.
- **Ambiguous and Contextual Semantics:** The language used to express the content of the tables is inherently versatile and ambiguous, as tables on the Web do not adhere to a controlled vocabulary. Often, the context of a table as well as external knowledge are essential to deduce the correct meaning.
- **No Quality Control:** The absence of quality control mechanisms means that the data is often incomplete and no guarantees for the correctness can be made. This results, for instance, in missing labels, duplicate tables or violations of integrity constraints.
- **No Formal Schema:** In contrast to database tables, tables on the Web do not provide a formal schema. There is no confirmation on data types, value domain constraints or referential constraints. The tables form a loose collection, only linked through the Web pages.
- **Small Individual Table Size:** The majority of tables exhibit only a very small number of rows and columns. Aside from some exceptions, these tables are smaller than average database tables. As a result, they frequently contain only a small, and not necessarily representative, sample of the individual attribute domains.
- **Vast Quantity:** The Web offers millions of distinct tables on Web pages and in linked documents. Similar to the Web in general, the number of tables is also constantly growing.
- **Variety of Domains:** The content of tables on the Web is not limited to specific domains. Instead, a table can cover any imaginable topic. As a result, domain-specific table recovery approaches are not sufficient to process such an open-domain corpus.

3.2 WEB TABLE APPLICATIONS

The volume, the domain diversity as well as the accessibility of the data have made tables on the Web an important resource for a number of applications. In this section, we introduce three prominent application areas studied in the literature, which all require table recovery in order to utilize the data embedded in Web tables. First, in Section 3.2.1, we take a look at the utilization of Web tables for the construction of domain-specific ontologies. Second, in Section 3.2.2, we investigate how *question answering* can benefit from the Web table data. And finally, in Section 3.2.3, we show how private databases can be augmented with public data from Web tables for *situational data analysis*. From these applications, we derive key requirements for the Web table recovery process, especially with regard to table understanding, in Section 3.2.4. Note that we focus on requirements for table understanding in particular, since detecting tables and recovering their cellular structure is already established as a mandatory prerequisite for all applications of table data.

3.2.1 Knowledge Extraction and Ontology Learning

A domain-specific ontology is a semantic data model, typically represented as a directed graph, that specifies individual concepts, their properties, and relationships between them within a domain of

knowledge or discourse. It provides a single, comprehensive view of the domain that is independent of lower level physical or logical data models specified by a system or application. Therefore, ontologies are frequently employed as an independent and consistent reference point. They enable the integration of data from multiple heterogeneous sources, as well as communication and interoperability between disparate systems (Liu et al., 2009). A prominent example is the *Gene Ontology* (Ashburner et al., 2000), which facilitates communication between researchers as well as the exchange of data and experimental results in molecular biology and genetics through a set of three ontologies and associated tools.

The *ontology construction process* still is, in many instances, a manual process that requires the assistance of experts with extensive knowledge of the respective domain. However, increasing demand for domain-specific ontologies has called for some level of automation of the process. The (semi-) automatic generation of ontologies is called *ontology learning*. A high-level view of the steps involved in this process is depicted in Figure 3.7. First, *terms* are extracted from the data source. These terms are then grouped to form *concepts*, which in turn are organized according to *relations*. These relations can be either taxonomic relations or non-taxonomic relations. Finally, relations can be generalized into *axioms*, which represent rules or constraints (Wong et al., 2012).

Tables are an important resource for ontology learning as they are often less ambiguous and, hence, easier to interpret than text. The *TANGO* project (Tijerino et al., 2003, 2005) is an example of a semi-automatic ontology learning approach based on the consolidation of information from multiple Web tables. Figure 3.8 shows the basic workflow, which involves three essential steps. First, a minimal ontology, the so-called *kernel ontology*, is designed by a domain expert or ontology engineer as the foundation of the target ontology. This ontology is generally very small, containing only the key concepts of the domain and some sample data. In a largely automatic process, the kernel ontology is then extended using table data. In the second step, a small local ontology is build from each table. An extensive table understanding process is employed to extract concepts, relationships as well as constraints, including key constraints and functional dependencies, from the table. In the third step, this local ontology is then merged into the kernel ontology, using a set of direct and indirect matching techniques, as well as user intervention to resolve conflicts. Steps two and three are repeated for each table, constantly expanding, correcting and consolidating the target ontology. The final result can then be applied to support a wide range of applications, including the conversion of current Web pages into Semantic Web documents (Tijerino et al., 2003).

In addition to generating new concepts or relations to add to an ontology, Web tables are also utilized for the task of *ontology population*, where *instances* of concepts or properties are collected. For example, for an ontology which contains the concept *City*, we may want to automatically add instances such as *London*, *Berlin* or *New York*. To extract instances from a table, first, each attribute of the table is matched to concepts or properties in the ontology. Then each instance value in the table that does not yet exist in the ontology is added and linked to its associated concept or property. Some tables describe more than one concept, in which case concept boundaries must be identified in order to associate properties with the correct concept (Syed et al., 2010). An example for ontology population is provided by J. Wang et al. (2012), where *Probase*, a very large probabilistic knowledge base, is extended with instances encountered in Web tables. Probase differs from a traditional ontology by not assuming correctness of the modeled concepts and relations, and, instead, acknowledging uncertainty. In general, in order to match a column or row in a table to a concept in an ontology, we require some overlap between the instances mentioned in the table and the ones already collected in the on-

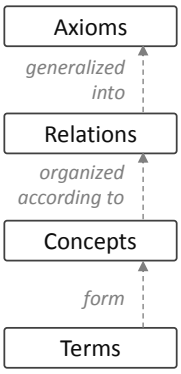


Figure 3.7: General process for ontology generation.

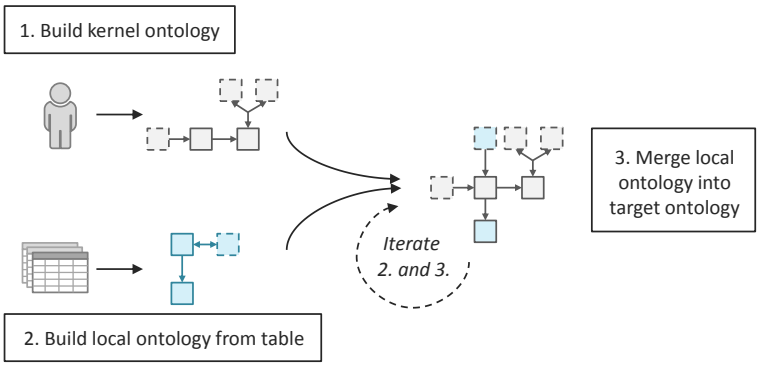


Figure 3.8: Ontology learning workflow in TANGO.

tology. Extending the knowledge base with additional data from Web tables enables more tables to be processed, as more evidence is provided to establish that the table describes the correct concept.

Requirements
From the processes involved in ontology learning and population, we can derive the following requirements or challenges for the table recovery process:

1. Building an ontology requires the identification of semantic concepts in the tables. This process is generally referred to as *conceptualization*. An important challenge is the identification of columns (or rows) that contain names or identifiers of entities, as they often serve as the *key attribute* for the concept. If multiple concepts are described in a table, we also need to identify concept boundaries in order to separate the individual concepts.

2. Associating non-key attributes with their respective keys requires the recovery of *functional dependencies* and *constraints* from the table and context.

3. In order to merge local and target ontologies, or to link table elements to an entry in an ontology, we need to match labels and entity mentions in the tables to the vocabulary used in the ontology. The inherent ambiguity of natural language, however, frequently causes a *vocabulary mismatch*, a common issue in data integration scenarios, which must be resolved.

3.2.2 Question Answering

The goal of *question answering (QA)* is the automatic retrieval of answers to user questions posed in natural language (Hirschman et al., 2001). It is very similar to *information retrieval (IR)*, the task performed by Web search engines, for example. However, instead of retrieving documents that might be relevant to the question, QA systems extract the actual answers from the documents. Hence, QA systems can be regarded as an extension to regular IR systems (Pinto et al., 2002). The main components of a QA system are illustrated in Figure 3.9. In a first step, the *question analysis*, the request is parsed and interpreted to identify its *type*, in order to determine the kind of answer that is expected

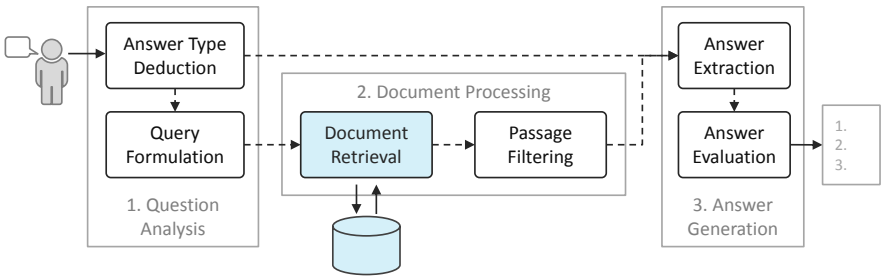


Figure 3.9: General process used for question answering.

by the user, as well as the appropriate procedure to extract this answer. Many QA systems support different types of questions, such as factual, opinion or summary questions (Hirschman et al., 2001). In the *document processing* step, documents that might be relevant to the question are then retrieved from a collection of indexed documents or a database. This is similar to the retrieval of documents or passages in an IR system. In the final step, the *answer generation*, all relevant documents are processed to extract potential answers. The retrieved candidates are ranked, often using the confidence of the system as the score, and the top-ranked answers are presented to the user.

As a data source, Web tables are especially suited for answering *factual questions* (sometimes also called fact lookup queries) (Yin et al., 2011). These questions often ask for the attribute of an entity, such as *"What is the capital of France?"* or *"What is the birth date of Nelson Mandela?"*. Facts like these are frequently found in tables on the Web.

Instead of indexing each table as a whole in the search index of the QA system, facts are extracted and indexed individually to facilitate an efficient search. Each fact consists of the content of a data cell, the content of all associated label cells (including key values), as well as any relevant context information mentioned in text outside of the table.

The *FACTO* system (Yin et al., 2011) is an example of a fact search engine. Facts are extracted only from *attribute-value tables* (see Section 3.1), vertically oriented tables with only two columns, and indexed as entity-attribute-value triples. As highlighted previously, this special type of table only holds information on a single entity per table. Therefore, the entity name is often factored out and only mentioned in the surrounding context. Before indexing the facts, additional extraction techniques are required to recover the missing context information. To match fact queries to the triples indexed by the system, synonyms and alternative entity names are considered, in addition to the original query terms. By replacing either the entity or the attribute with an alternative term in the query, multiple queries are generated and the returned results combined.

Another example is *QuASM*, a QA system that operates on different types of structured documents on the Web, including tables encoded in text and HTML (Pinto et al., 2002). In contrast to FACTO, this system is not limited to a specific table layout. Facts are, again, extracted from the table before indexing, consisting of cell data associated with column and row headers as well as document titles. Heuristic rules are used to identify the indexing scheme in the tables. An evaluation of the system achieved reasonable results, but also indicated that the quality of row and column labels directly influences QA performance.

The *QEWT* system answers open-domain quantity queries, using numerical data found in Web tables (Sarawagi et al., 2014). Compared to textual data, numerical data is more likely to exhibit *natural variation*, meaning that additional parameters such as units, scale factors or time constraints affect the values in the tables. For instance, the answer to the question *"How large is the average revenue of Microsoft?"* changes depending on the currency or the timeframe considered. Evidence for these constraints are frequently found in attribute labels or in the context, such as the title or caption. However, for some tables, no such evidence is available. To answer quantity queries with uncertainties, the QEWT system uses collective inference to combine answers from multiple sources and returns a list of potential value distributions to the user.

Requirements
In addition to correctly identifying semantic concepts and resolving semantic ambiguities in the tables, question answering systems also highlight the following requirements for Web table recovery to ensure the validity of the extracted facts:

1. Attributes with descriptive labels improve the chances for the correct answers to be identified.

2. Values with natural variation, especially numerical data, depend on additional contextual constraints. Recovering these parameters is important to ensure that the data is interpreted and used correctly.

3.2.3 Entity Augmentation and Situational Data Analysis

Situational Data Analysis is an emerging form of data analysis that facilitates the processing of ad-hoc queries to satisfy an unexpected information need. Common scenarios that require situational analysis include exploratory search and what-if analysis. Given the unexpected nature of the information need, in addition to pre-existing data, ad-hoc queries often require additional data that does not yet exist in the available databases or data warehouses and needs to be added to carry out the analysis. Traditionally, integrating data from multiple disparate sources requires an extensive *ETL* process (short for Extract-Transform-Load) to convert the data into a uniform and consistent data collection. However, in many cases, the information need is only temporary and many ad-hoc queries are executed only once, which often does not justify initiating such a costly integration process. Therefore, situational data analysis attempts to reduce the integration overhead and perform data search and integration on-the-fly.
To collect the missing data, an ad-hoc *information gathering* task is required. We distinguish two types of information gathering, as illustrated in Figure 3.10. *Entity expansion* adds new instances (i.e. new rows in a relational table), given the schema or sample instances. In contrast, *entity augmentation* extends a table by a single or multiple attributes and their respective values (i.e. new columns). Web tables are a rich resource for these information gathering tasks, as they are freely available and cover a wide range of entities and topics. Figure 3.11 shows the process involved in augmenting a local database with data from Web tables, which consists of three main steps. In the *query processing* step, the list of entities E that are to be augmented with the new attribute A is retrieved from the local database. Both, the entities and the label for the new attribute are then used to search for the missing attribute values in the *information gathering* step. To efficiently identify relevant candidate tables, the Web tables are generally preprocessed and indexed in a designated retrieval system. Schema matching

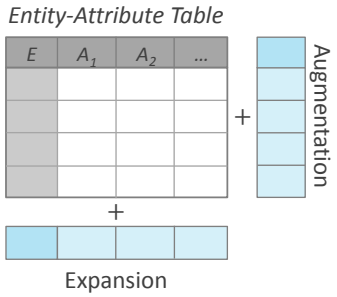


Figure 3.10: Main types of information gathering.

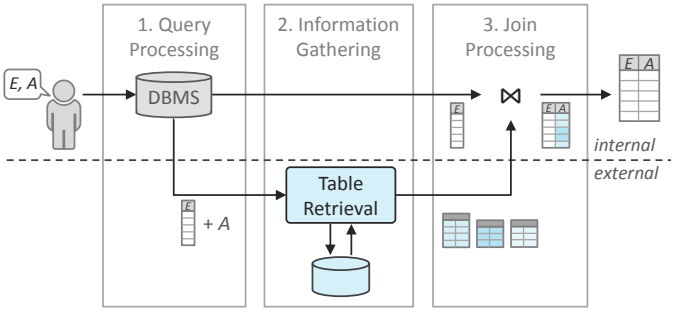


Figure 3.11: General process used for entity augmentation using Web tables.

and entity resolution are employed to match the entities E and label A to the candidate tables. Finally, in the *join processing* step, the relevant values from the retrieved candidate tables are consolidated and joined to the original table in the local database. A consolidation of values from different sources is often necessary, as it is unlikely to find a single table that covers all entities in E. The augmented table can then be used for further analysis.

An example for entity augmentation using Web tables is provided by the *OCTOPUS* system developed by Cafarella et al. (2009). The system uses the *EXTEND* operator, which takes as input a column and a keyword describing the missing attribute, to augment tables. Web tables are clustered to enable all similar tables that are about the missing attribute to be found quickly.

Another example system is *InfoGather*, developed by Yakout et al. (2012). An addition to attribute labels, InfoGather enables the use of sample instances to describe the missing attribute A and guide the integration process. To efficiently find all relevant candidates, all Web tables are preprocessed by splitting them into so-called *Entity-Attribute binary relations* and integrating all of these binary relations into a large matching graph. In the information gathering step, this graph is utilized to efficiently find all relevant tables, including direct and indirect matches. The authors highlight that including indirect matches, i.e. corresponding attributes that do not match the attribute label used for the search, significantly increases precision and coverage of the entity augmentation process. To match binary relations extracted from the Web tables, a variety of features are compared, including entity coverage, attribute labels and the table context.

Requirements

Utilizing Web tables for situational data analysis shares many requirements and challenges with other Web table applications, including the identification of concepts described by a table, the correct association of attributes with their respective key attributes, as well as techniques to resolve semantic ambiguity in labels and entity mentions. An additional requirement that arises especially in the context of entity augmentation is the need to effectively identify *related tables*, in order to increase precision and coverage despite non-matching attribute labels.

3.2.4 Analysis of Application Requirements

In the previous sections, we introduced three application scenarios for tables extracted from the Web: ontology learning, question answering and situational data analysis. For each application, we derived a set of requirements regarding the table understanding process that are necessary in order to enable the correct and efficient use of Web table data. While some key requirements are shared by all three applications, other requirements are unique to individual use cases. The following list summarizes the requirements:

1. **Conceptualization:** A correct utilization of the data extracted from Web tables requires the identification of semantic concepts described in the tables. This includes locating concept boundaries as well as identifying (natural) key attributes.

2. **Recover Functional Dependencies and Constraints:** In order to identify concept boundaries, individual attributes must be associated with their respective key attributes.

3. **Address Vocabulary Mismatch:** Linguistic heterogeneity and ambiguity in the attribute labels as well as entity names need to be resolved in order to integrate the data semantically.

4. **Use Descriptive Labels:** To ensure that the table data can be identified and utilized by applications, labels that provide an adequate description of the data are desirable.

5. **Recover Contextual Constraints:** Especially numerical attributes, but also categorical attributes, are often contextual, i.e. they depend on additional dimensions, such as time. Tables, however, are in most cases not self-contained and such contextual dimensions are often not part of the table itself, but the surrounding text or metadata.

6. **Find Related Tables:** As a whole, Web tables cover a wide variety of topics and entities. However, individual tables are often very small and several related tables must be combined to retrieve all the needed information.

3.3 REVISITING TABLE RECOVERY

In Section 2.2.1, we illustrated the process involved in recovering the relations underlying tables in general. After analyzing the characteristic properties of Web tables in particular, as well as the specific requirements of key applications that utilize and benefit from Web tables, we can now specify the recovery process for the special case of Web tables. In the following sections, we describe the main components of the process and present related work from the literature. We put the main focus on Web table understanding, as it is the most challenging part of the process and also the subject of this thesis.
Many significant contributions have been made to recover meaningful data from Web tables. However, it is still an open challenge, as many of the contributions only have a limited scope. After presenting related work, we highlight the most prominent limitations and outline how we address these limitations in order to improve Web table understanding.

3.3.1 Table Recognition

In the context of Web tables, techniques to detect tables in a document and to recover the physical table structure are less prominent in the table recovery pipeline, compared to printed documents. The majority of tables on the Web appear in formats that inherently support table structures, such as HTML, CSV and spreadsheet formats. Therefore, the research focus in the literature is mainly on these formats, although other formats exist on the Web, as well.
In HTML files, tables are generally distinguished from other content via the designated `<TABLE>` tag and structured using, among others, `<TH>`, `<TR>` and `<TD>` tags. These tags enable a straightforward detection and extraction of HTML tables. The main challenge for table recognition in Web pages is the identification of tabular structures that do not represent genuine tables and are used for layout purposes, instead. H.-H. Chen et al. (2000) propose two heuristic rules based on the table size and the fraction of images or hyperlinks in the table to filter out unlikely candidates. Y. Wang et al. employ statistical classification to detect genuine tables, using a feature set containing layout features, content type features and word group features (Y. Wang et al., 2002). The proposed approach achieves very accurate results, with an F_1 score of over 95% reported for a corpus of several thousand tables. Cafarella et al. translate HTML table classification to a larger scale, using a rule-based classifier to identify $\approx 150M$ genuine tables (Cafarella et al., 2008b).
Relying on designated markup to identify tables represents the most common detection approach. However, alternative techniques exist, as well. For example, Gatterbauer et al. (2007) perform table detection and structure recognition based on visual clues in the document. From the DOM tree of the Web page, they drive so-called *visualized element nodes* (VEN) and analyze their alignment to detect tabular structures. Another alternative is the detection of tables of interest based on *extraction ontologies*, proposed by Embley et al. (2005). However, these often manually created ontologies are domain and application specific and do not present a generalized extraction approach.

3.3.2 Table Understanding

Automated Web table understanding is an evolving process that encompasses many different processing steps. Figure 3.12 illustrates the main steps involved. For each step, we give an overview of the most relevant research contributions in the literature. Note that in some cases, several processing steps are considered in combination, instead of separately, if they strongly influence each other. Especially header recovery and conceptualization are often performed collectively.

Table Classification
Recovering the logical structure of Web tables, i.e. differentiating between label and data cells and recovering the relationships between them, is a challenging task, due to the great variety of layouts. A first approach to address this heterogeneity is to classify tables into broad categories based on the layout type. Two classification schemes are proposed by Crestan et al. (2011) and Lautert et al. (2013), both relying on a combination of lexical and layout features to inform the classification. A classification of tables by layout type enables a more accurate analysis, as algorithms can be tailored to each layout category.

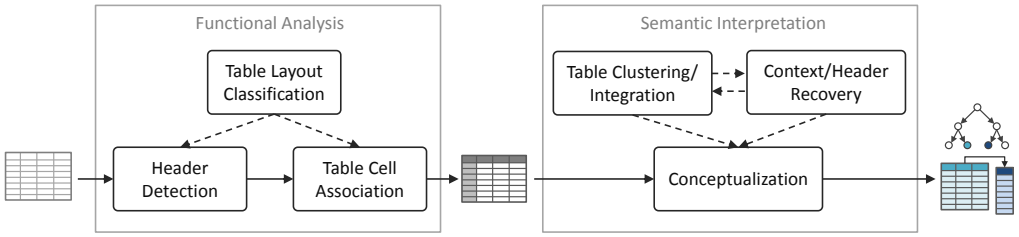


Figure 3.12: Overview of the process involved in Web table understanding.

Header Detection

To detect label cells in Web tables, a number of different techniques have been proposed. The contributions by Yoshida et al. (2001) and Tengli et al. (2004) rely on explicit terms encountered in tables to distinguish between labels and values. Learning from a set of example tables, the algorithms infer how likely it is for certain terms to appear in label and data cells, respectively. Based on this classification, the most likely position for the row and/or column headers are inferred. Pinto et al. propose simple heuristics for the identification of headers, essentially expecting the first row and the first column of a table to contain label cells (Pinto et al., 2002). Cafarella et al. use a rule-based classifier to detect headers in simple, relational-style Web tables (Cafarella et al., 2008b). Expecting the header to be positioned in the first row of a table, the classifier distinguishes between tables with and without header. More complex indexing schemes are analyzed by Z. Chen et al., who propose an approach to identify the so-called *data frame* in spreadsheet tables (Z. Chen et al., 2013). The data frame separates the boxhead, stub and body sections in the table. Using a conditional random field (CRF) approach, each row is labeled (as *title*, *header*, *data* or *footnote*) and the data frame is derived from these labels.

Cell Association

For tables with simple, two-dimensional indexing schemes that do not feature nested or hierarchical headers, the association between label and data cells is straightforward. Each data cell is simply associated with the labels in the corresponding row and column headers. For more complex indexing schemes, the structure of the header must be resolved first. Z. Chen et al. introduce a classification approach to identify and resolve hierarchical attributes in spreadsheets (Z. Chen et al., 2013). As an alternative, Seth et al. apply a rule-based cutting algorithm to resolve hierarchical attributes (Seth et al., 2010).

To associate data cells with their corresponding labels in a complex hierarchical indexing scheme, Z. Chen et al. utilize a collective inference technique (Z. Chen et al., 2014). Using an undirected graphical model, they identify the most likely parent-child pairs, both between labels in the hierarchy as well as between label and data cells. The proposed approach also supports user intervention to correct any wrong associations interactively.

Table Clustering and Integration

An important strength of Web data in general is the amount of data that is available. Considering the data in its entirety gives a realistic reflection of the expressiveness and ambiguity of natural language and provides a basis for statistical and collective inference techniques (Halevy et al., 2009). These

can be used to address vocabulary mismatch or to recover implicit information. However, in order to utilize a large collection of Web tables efficiently, we require some form of integration or clustering. To identify related tables, different notions of *relatedness* between tables have been proposed.
Yoshida et al. cluster Web tables into groups that represent the same type of objects by identifying unique attributes that are characteristic for each type (Yoshida et al., 2001). After evaluating the uniqueness of each distinct attribute in the corpus, the most prominent attributes are selected and tables clustered, if they share one of these attributes. The effectiveness of the uniqueness score strongly depends on the size and coverage of the table corpus. A related approach that links tables based on shared labels is described by Cafarella et al. (2008a). Here, relatedness is measured using the *join neighbor similarity*, which takes schema coherence into account to distinguish between different meanings of the same label. Only tables where the shared label is likely to have the same meaning are clustered using an agglomerative clustering approach.
In the *OCTOPUS* system, clustering of tables is achieved using various similarity measures, based on term frequencies and term length statistics (Cafarella et al., 2009). The goal is to find tables that can be combined through a union operation. However, the very simple similarity measures do not take the logical structure of the tables into account to support this goal. In contrast, Das Sarma et al. define the concept of an *entity complement* to identify tables that can be combined through a *union*. Similarly, they define a *schema complement* to find tables that can be combined through a *join*. Several complex similarity, consistency and expansion scores are used to find related tables, taking the logical structure of the tables into account (Das Sarma et al., 2012).
The most comprehensive integration approach is implemented in the *InfoGather* system (Yakout et al., 2012), which combines direct and indirect table matching techniques. The system exclusively focuses on binary tables consisting of an entity column and an attribute column. The similarity between binary tables is established directly via similarity scores based on value overlap, matching labels and context similarity. In addition, similarity is established indirectly via the *topic sensitive page rank* approach, which propagates similarity scores through the graph of binary tables. Indirect matching enables the identification of tables that are conceptually related, but do not share the exact same labels or entity mentions.

Context and Header Recovery
The recovery of missing or implicit information in Web tables involves different types of information: (1) descriptive attribute labels, (2) units and scale factors for numerical attributes, and (3) constant attributes representing contextual constraints that have been factored out to the context. To recover the relevant information, we can take into account the context of the table, other related tables or external sources of commonsense knowledge, such as *Probase* (Wu et al., 2012), that contain general information about real-world entities, their types and relationships. Table 3.3 organizes related work according to these categories. Additional attributes, such as temporal or spatial constraints, are often found in the table context. To identify and extract these attributes, Cafarella et al. propose a special *CONTEXT* operator for the *OCTOPUS* system (Cafarella et al., 2009). Significant terms with a high *tf-idf* score are selected from the context as candidate attributes. In addition, related tables that describe the same entities and attributes are considered to find mentions of these additional attributes. Values that appear in the context as well as in related tables are considered to be potential hidden attributes of the entities in the original table. An alternative approach is proposed by Ling et al. (2013). Here, several tables with the same schema, which appear on related Web pages, are

Table 3.3: Overview of related work addressing the recovery of missing, incomplete or implicit information.

	Context	Related Tables	External Knowledge
Attribute Labels		Yakout et al. (2012)	Limaye et al. (2010) Venetis et al. (2011) Mulwad et al. (2011) J. Wang et al. (2012)
Units & Scale Factors		Zhang et al. (2013) Sarawagi et al. (2014)	Sarawagi et al. (2014)
Hidden Attributes	Cafarella et al. (2009) Ling et al. (2013)	Zhang et al. (2013) Ling et al. (2013)	

considered. The text sections surrounding the tables on the Web pages are assumed to be very similar. Hidden attributes in the context are discovered by aligning the text segments from different pages and searching for differences. To find appropriate labels for the added attributes, an external knowledge base containing *isA*-relations is utilized.

In the *InfoGather* system, a graph of related tables is employed to derive new column labels or synonyms for existing column labels (Yakout et al., 2012). The labels of related tables are clustered based on attribute value overlap to identify synonyms. Labels are then propagated through the graph to annotate columns without a useful label. In *Infogather+*, the concept of propagating information is extended to numeric attributes (Zhang et al., 2013). Units, scale factors and time constraints are propagated as column annotations from tables that provide the information to related tables that are missing these details. The system only considers information provided in the headers of tables, not in surrounding context. An alternative approach to recover units and scale factors for numeric attributes is presented by Sarawagi et al. (2014). A catalog of common units and scale factors, which provides a full name, symbols and name variations for each unit, is used to annotate the attributes. In addition, co-occurrence statistics, collected from a large collection of related tables, are taken into consideration to identify the most likely unit type per attribute and, thus, reduce the search space of potential units.

To recover descriptive attribute labels for columns with missing or non-informative labels, several techniques have been proposed that utilize external information. In specialized or general-purpose knowledge bases, information about concepts, their attributes and relationships are collected. In addition to the description of concepts, they also contain a large collection of instances, i.e. entities that are associated with these concepts. Given a column that contains several entity names, a knowledge base can be utilized to infer the most likely concept for these entities, which serves as the new column label. J. Wang et al. (2012) employ the general-purpose probabilistic knowledge base *Probase* (Wu et al., 2012) to infer the most likely concept per column. Probase has been extracted automatically from a large collection of Web data and provides extensive coverage of concepts and associated attributes.

Venetis et al. use a specialized taxonomy of *isA*-relations to annotate columns (Venetis et al., 2011). Similar approaches are also presented by Mulwad et al. (2011) and Limaye et al. (2010), where the external knowledge bases *Wikitology* (Syed et al., 2011) and *YAGO* (Suchanek et al., 2007) are used, respectively. Comparing these various approaches highlights that the effectiveness of utilizing an external source to infer attribute labels strongly depends on the size and coverage of the knowledge base.

Conceptualization

The goal of conceptualization is to understand what a table is about, which involves recognizing entities, semantic concepts with associated attributes and relationships in the data. In general, the inferred semantics can be specified *explicitly* or *implicitly*. An explicit specification is achieved by linking tables or table elements to entries in a reference ontology or knowledge base. In contrast, an implicit specification is provided by clustering or integrating similar tables, so that a mediated schema of the related tables defines the concepts and relationships involved.

Explicit annotation techniques have been studied extensively. They are very similar to the techniques used to recover descriptive attribute labels, so that attribute recovery and conceptualization are often performed collectively (see (J. Wang et al., 2012) or (Mulwad et al., 2011)). In addition to utilizing different knowledge bases for inference, the approaches also differ in the way inference is carried out. One group of algorithms follows a *successive labeling* approach. First, individual table sections are labeled separately, in some cases with multiple possible annotations. Afterwards, the individual annotations are combined to select the final labels, which represent the most probable interpretation of the table. An important part of these labeling approaches is the detection of *entity columns*, which contain the names or identifiers of real-world entities and often serve as a concept's key attribute. Venetis et al. (2011) propose two techniques, a rule-based approach and a learning-based approach to identify entity columns, with the latter achieving a higher accuracy. Both approaches are independent of the actual labeling approach. In contrast, J. Wang et al. rely on the external knowledge base to detect entity columns (J. Wang et al., 2012). The most likely entity column is the one where the concept of the column has the strongest association with the other attributes in the table, based on the information stored in the knowledge base. Both entity column detection techniques assume tables that describe a single semantic concept, meaning that all attributes in the table are regarded as attributes of the same concept.

The second group of algorithms uses *collective inference* to interpret Web tables. Utilizing probabilistic graphical models, these approaches infer the most probable interpretation by considering all elements, i.e. entities, attributes and relationships, at the same time. Limaye et al. (2010) use such a probabilistic graphical model to annotate Web tables with labels extracted from the *YAGO* knowledge base. A similar approach is also proposed, but not implemented or evaluated, by Mulwad et al. (2011). In comparison, collective inference often achieves more accurate results, but is also much more expensive to compute compared to successive labeling techniques.

All annotation techniques mentioned so far require the entities mentioned in the table to be included in the external knowledge base. However, in many cases, it is very unlikely to find all entity names in the knowledge base, as they often only contain the most important or most representative entities for each concept. Quercini et al. address this fact by proposing an annotation technique that relies on Web search instead, which is more likely to produce information on less well known entities (Quercini et al., 2013). For each entity mention in the table, several text snippets are collected via text search. Using a classifier trained on text features that are characteristic for a given set of concepts, these text

snippets are classified to identify the concept of each entity. The most likely concept of all entities is then inferred via majority voting.
Implicit conceptualization is achieved through table clustering and integration, and many of the techniques introduced in the previous paragraph contribute to this task, including (Cafarella et al., 2008a) and (Yakout et al., 2012). An alternative approach, which is based on database normalization, is proposed by D. Z. Wang et al. (2009). First, a collection of Web tables is integrated into a single mediated schema, which is then normalized. Instead of traditional normalization techniques, the authors propose *semantic normalization* in order to split the large schema into smaller, coherent schemas that represent semantic concepts and relationships. Exact functional dependencies, which are necessary for database normalization, are generally not available for Web tables. Therefore, *probabilistic functional dependencies* are inferred from a single or multiple tables.

3.3.3 Simplifying Assumptions

As highlighted in the review of related literature, many significant contributions have been made towards automated understanding of Web tables. However, the overall task is very challenging due to the complex dependencies between table structure, content and context, interlaced with structural and linguistic ambiguity. To address this complexity and uncertainty, a number of *simplifying assumptions* have been proposed. By addressing only tables with specific characteristics, these assumptions narrow down the complex problem space, so that individual sub-tasks become manageable. The most prominent assumptions are listed below:

- **Simple Relational Table Structure:** To reduce the structural ambiguity, many approaches only consider Web tables that feature a layout similar to relational tables. These tables list attribute values vertically, with attribute labels positioned in the column header and no designated stub section. As illustrated in Section 3.1, however, tables with this type of layout represent only a fraction of the tables available on the Web.

- **Single Concept per Table:** Web tables are frequently assumed to be semantically normalized, meaning that each table describes only a single semantic concept. As a consequence, each attribute in the table characterizes the same concept, with all non-key attributes determined by the key of the table. This simplification makes it easier to infer the relations underlying the data. It is particularly valuable when attributes are matched to entries in an external knowledge base, in order to infer the concept of the table.

- **Simple Keys:** In addition to assuming that each table describes only a single concept, the complexity of the relation represented in a table is often further reduced by discarding all tables with compound keys. It is assumed that, for each table, there is a single attribute (usually a name or identifier), that uniquely identifies each entry within the table. While so-called *named entities*, such as persons, companies or geographical entities often have such a simple key, other entities, especially inanimate or abstract objects, often do not have a designated name and require a compound key.

- **Loose Definition of Context:** The importance of including context in the table recovery process is well established. Generally, surrounding text, captions and footnotes are considered as

potential sources. However, establishing which parts of these sources are actually relevant in order to understand the meaning of a table is a challenging task and, therefore, often excluded or only considered marginally. As a result, many approaches either take all context into account or do not consider any context at all.

These simplifying assumptions enable the development of table understanding techniques by limiting the number of influencing factors, thus isolating individual challenges in the understanding process. However, the resulting techniques are limited in scope and can lead to inaccurate results, if the assumptions do not hold.
In this thesis, we address these limitations of previous work by extending the Web table recovery, and, especially, the Web table understanding process with techniques that enable these assumptions to be relaxed.

3.3.4 Outline

In the subsequent chapters, we propose extensions to three areas of Web table understanding, as highlighted in Figure 3.13: (1) table layout classification, (2) context and header recovery, and (3) conceptualization. In detail, we study the following aspects:

- **Chapter 4:** First, we take a look at the main table layouts that are used in Web tables and present features that are suited to identify these layout categories. We then investigate two alternative approaches to incorporate table layout classification in the table understanding process.
- **Chapter 5:** In this chapter, we study the role of contextual information in Web table understanding. We propose a technique to estimate the relevance of context information and reduce long, noisy context sections to relevant paragraphs. Furthermore, we utilize the table context to extract attribute-specific annotations that provide a richer description of the table content.
- **Chapter 6:** Finally, we address the frequent assumption that tables on the Web only describe a single semantic concept. To relax this assumption, we describe a semantic normalization algorithm that decomposes tables describing multiple concepts into several single-concept tables.

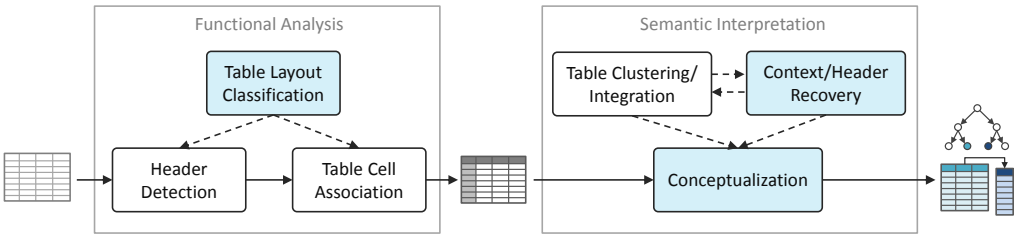


Figure 3.13: Areas in the Web table understanding process that we extended in this thesis.

4

TABLE EXTRACTION AND CLASSIFICATION

4.1 Related Work

4.2 Classification Problem

4.3 Feature Specification

4.4 Experimental Evaluation

4.5 Summary and Discussion

In the previous chapter, we analyzed the characteristics of tables available on the Web. One of the most prominent features of these tables is the diversity of the table structures and layouts. As pointed out before, the same data can be displayed in tables in many different ways, depending on the type of information that is presented, the purpose of the table, restrictions of the Web page layout, or simply design preferences. Therefore, different table layouts can be identified on the Web, where especially the position of the label cells as well as the alignment of attribute values vary.
Some of the information presented by a table is implicitly conveyed through its structure and layout, due to its compact representation of the data. Correctly recognizing the layout is important in order to understand the role of each table cell, distinguishing between label and data cells. Moreover, recognition of the layout enables locating attribute and entity names as well as additional categorical dimensions in the label cells of a table and associating them with entries in the table's body. From these *reading paths*, i.e. associations between label and data cells, we can start to recover the relation underlying the table (Hurst, 2000).
In Section 3.1, we distinguished three main classes of table layouts, based on the alignment of attribute values: (1) *vertical listings*, (2) *horizontal listings*, and (3) *matrix tables*. Each of these classes can be subdivided further, based on additional criteria, such as the semantics or dimensionality of the data displayed. Defining a single table model to recognize and process tables with such varied layouts is very complex and challenging. Therefore, in order to reduce the structural diversity and ambiguity, various approaches that process Web tables only focus on tables with a layout similar to relational database tables (Cafarella et al., 2008b). These tables, which represent vertical listings in our classification scheme, feature attribute labels in the column headers and no designated stub section. A simple, uniform model can be used to describe and process these tables, for instance to detect headers (Cafarella et al., 2008a) or extract binary relations (Yakout et al., 2012). Other subsequent table recovery tasks, such as context extraction or conceptualization, also rely on this model of the table structure. Assuming a single uniform layout clearly limits the range of Web tables that can be processed and utilized for question answering or fact search. Identifying different layout types makes it easier to specify a model for each table, which can be utilized in subsequent processing steps.
In addition to identifying different table layouts, we face another challenge, especially in the context of tables embedded in Web pages. While HTML documents provide support for tabular structures with designated `<TABLE>` tags, the same structures are also frequently used for other purposes, such as the spatial arrangement of page content. Instead of relational data, these *layout tables* contain other document content, such as images or menu items. Figure 4.1 shows an example Web page, where tables are utilized to arrange the position of images. The vast majority of HTML tables in Web pages are layout tables instead of genuine data tables (Cafarella et al., 2008b).
Identifying genuine tables and discriminating between different table layouts each represent important aspects of Web table recovery. Both tasks can be regarded as classification tasks, yet each with a very different objective.
In the literature, we can identify two alternative approaches for incorporating these tasks. Some authors, such as Cafarella et al. (2008a), consider genuine table classification as a separate task, while others incorporate it into table layout classification by simply considering layout tables as another category (Crestan et al., 2011; Lautert et al., 2013). Combining both aspects into a single task is motivated by the fact that for both tasks similar table features can be utilized, such as the presence and location of a header or the consistency of cell entries. However, the different objectives and the fact that layout tables are significantly more frequent than any other class of tables, especially in Web

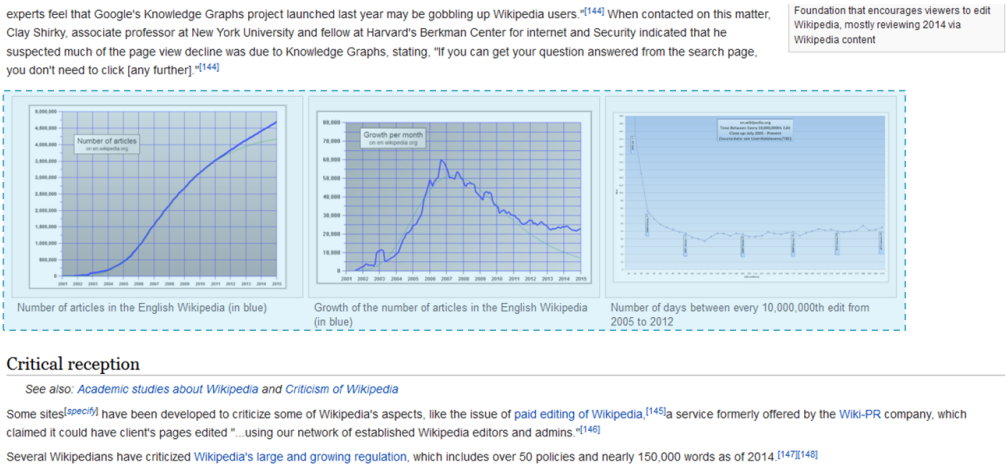


Figure 4.1: Example Web page that utilizes HTML tables to arrange images [1].

pages, suggest that a separate processing of these two tasks is preferable. Table detection has been carried out successfully, both as a separate task as well as combined with table layout classification. However, these approaches have not been compared directly and, so far, layout classification has not been studied as a separate task.

In this chapter, we address the two alternative approaches to incorporate table detection and layout identification, and provide a comparison to determine which approach achieves a more accurate prediction. We collect and consolidate various table features proposed in the literature and evaluate different classification algorithms in order to come to a reliable conclusion.

In detail, this chapter is organized as follows. First, we review related work in the field of table detection as well as table layout classification for Web tables (Section 4.1). We then formally define the classification problems, including the classification schemes, that we address in this chapter (Section 4.2). Following this general description of the classification tasks, we take a closer look at suitable features of Web tables. A wide range of features defined over the whole table as well as smaller subsets are employed to facilitate an accurate classification (Section 4.3). In an experimental evaluation on real-world data, we evaluate the effectiveness of the selected features. We compare various state-of-the-art classification algorithms regarding their suitability to the task and, ultimately, their classification performance (Section 4.4). Finally, we conclude this chapter with a summary of our findings (Section 4.5).

4.1 RELATED WORK

In this section, we review previous work proposed in the literature that addresses either the detection of genuine tables or the identification of table layouts. We mostly focus on work that specifically targets tables on the Web, which have received significant attention by researchers in recent years.

Table detection and layout recognition have also been studied in the context of document tables in

[1] Source: `https://en.wikipedia.org/wiki/Wikipedia`

general, especially tables in plain text files, for instance by Ng et al. (1999) and Pinto et al. (2003). However, the challenges of locating and extracting tables from plain text differ substantially from the detection and analysis of HTML tables. While HTML markup discloses the location of tables as well as the table's cell structure in Web pages, in plain text, table and cell boundaries must be derived from surface features, such as line art, white space or special punctuation. A more detailed review of table detection in text documents is provided in Chapter 2. Here, we focus on the specific challenges that arise in the context of Web pages.
A number of table detection and analysis approaches have been proposed that specifically target tables on the Web. These approaches take the specific characteristics of Web tables into consideration and often utilize features specifically found in Web pages, such as the document object model (DOM) representation of a table.

Genuine Table Detection
The detection of genuine Web tables has received significant attention from the research community, mainly focusing on tables embedded in HTML documents. In contrast to table representations in plain text documents, HTML tables separate the content from visual features such as lines and whitespace. Consequently, alternative features are required to characterize genuine Web tables for identification.
H.-H. Chen et al. (2000) address the detection of HTML tables by proposing a set of heuristic rules and cell similarity measures that distinguish genuine tables from tables used for layout purposes. These simple rules eliminate tables with less than two cells as well as tables that contain a significant amount of hyperlinks, forms or figures. For a test set of roughly $3,000$ tables, about 58% of tables are filtered out by these rules as non-genuine tables with an accuracy of 98.9%. For the remaining tables, a set of cell similarity measures that evaluate the consistency of table content are used to filter out any remaining layout tables. For neighboring cells, the similarity of the cell content is measured based on content type or string similarity. The total number of neighboring cells regarded as similar determines whether a table is a genuine relational table or a layout table. An overall F-measure of 86.5% is reported, with the most significant contribution achieved by a similarity measure detecting numeric content. Overall, H.-H. Chen et al. achieve reasonable results with very simple measures.
Similar heuristic rules are implemented by Penn et al. (2001) for the detection of genuine Web tables. A table is regarded as genuine, if it is a leaf table (i.e. it does not contain another table in a cell), contains multiple rows and columns, and the size of each table entry is below a predefined threshold. Furthermore, genuine tables do not contain lists, forms, images or other non-text formatting tags. For a small test corpus, this detection approach achieves an F-measure of 88.01%. The authors also point out that syntactic and semantic coherency within the rows or columns of a table are important characteristics to identify genuine tables. However, no specific measures for coherency are proposed and these characteristics are not included in the detection algorithm.
Y. Wang et al. are the first to apply machine learning techniques to the detection of genuine Web tables (Y. Wang et al., 2002). Decision trees as well as support vector machines (SVM) are considered for the task. A large set of features, including structural features and content type features, are utilized. In addition, a *word group feature* is proposed that treats tables as text documents and genuine table detection as a document categorization problem. Based on its content, tables are categorized as either genuine or non-genuine, using state-of-the-art text categorization techniques, including a vector space approach, Naive Bayes and k-nearest neighbor classification. An overall evaluation re-

ports a maximal F-measure of 95.88% for decision trees and 95.89% for support vector machines with a radial base function (RBF) kernel, suggesting that the choice of learning algorithm has only little impact on the classification accuracy. The results further indicate that the structural and content type features contribute much more than the complex word group feature, with an F-measure of 95.73% achieved using only structural and content type features. Finally, the machine learning-based approach is compared to the rule-based approach of Penn et al., which achieves an F-measure of 87.63% on the same corpus. The learning-based approach clearly outperforms the simple heuristic rules.

Cafarella et al. combine simple rules and statistical classifiers to detect relational tables in a huge corpus of 14.1 billion HTML tables extracted from a Web crawl (Cafarella et al., 2008b). The amount of genuine tables is estimated to be only 1.1% of the entire corpus, however attribute/value tables are not included, as they are not regarded as relational tables. Similar to previous techniques, the majority of obviously non-relational tables (89.4%) is eliminated using a set of simple heuristic rules. The remaining tables are classified using rule-based classifiers trained on 7 simple features inspired by the work of Y. Wang et al. (2002). These features address the table structure as well as the consistency of the content. Tuning their classifier to maximize recall at the prospect of loss of precision, Cafarella et al. report an average precision of 69% and an average recall of 84% for a subset of the corpus, consisting of several thousand tables. The rules and features utilized in this approach do not present novel contributions. However, Cafarella et al. are the first to apply a table detection algorithm to a corpus at Web scale.

A significantly different classification approach is proposed by Son et al. (2013), who utilize the structural information provided by the DOM tree of a Web page to detect genuine Web tables. The DOM tree of the table as well as the DOM tree of the surrounding document are directly used as features in the classification tasks. A specialized parse tree kernel is proposed for SVM-based classification. Additionally, the content type features proposed by Y. Wang et al. are used to incorporate content coherency. On a test corpus of several thousand tables, Son et al. report an F-measure of 98.58% for an SVM-based classification combining structural and content type features. This approach outperforms any previous technique by incorporating the structural characteristics of the DOM trees. While previously proposed features were mostly applicable to tables independent of the file format, the features proposed here are specific to HTML tables.

Table Layout Classification

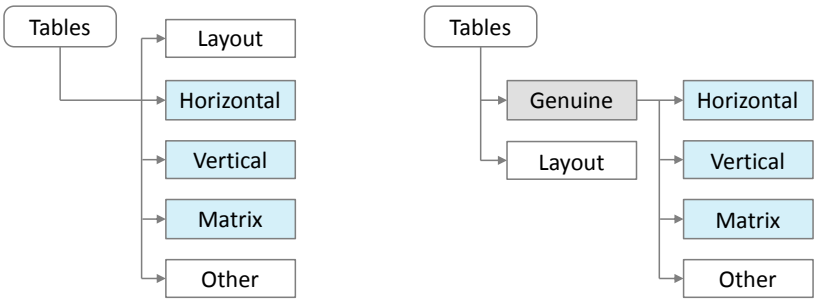
In addition to distinguishing between genuine and non-genuine tables, more fine-grained classification schemes for Web tables have been proposed, which take the layout and structure of the tables into account.

The first classification scheme has been proposed by Crestan et al. (2011). At the highest level, Web tables are categorized as either *relational knowledge* or *layout* tables. Relational tables are further divided into seven categories: listings (vertical and horizontal), attribute/value tables, matrix tables, calendars, enumerations (i.e. lists), and forms. Layout tables are divided into two categories: navigational tables and formatting tables. Consequently, the classification task assigns each table to one of ten categories, with the category *Others* added for all tables that do not fit into any of the nine specified categories. A wide range of features characterizing the structure and cell content are used to classify the tables. Features are considered for the table as a whole (global features) as well as for individual columns or rows (local features). In particular, the first two rows and columns as well as the last row

and column of a table are considered for local features. Before classifying the tables, Crestan et al. apply a simple rule-based filter to eliminate tables that are obviously not relational. The applied rules are similar to those proposed by Penn et al. (2001), and filter out tables with less than two rows or columns, and cells with more than 100 characters. The authors report a reduction by more than 80%, with 93% of the eliminated tables identified as layout tables. The remaining tables are classified using a *gradient boosted decision tree* model, which represents an ensemble of decision trees to avoid overfitting. The features and classifier are evaluated using 20-fold cross-validation on a hand-labeled corpus of $5,000$ tables. The classifier achieves an overall accuracy of 75.2%, with the best results achieved for formatting tables, calendars, attribute/value tables and vertical listings. The classification results indicate that lexical features computed per column or row, such as the ratio of string or numerical content, are the most effective features, especially for vertical listings and attribute/value tables. Furthermore, the authors point out a significant amount of misclassifications between attribute/value tables and listings. Attribute/value tables can be regarded as a special case of horizontal listings with only two columns. The classification results suggest that they should be regarded as one class instead of two, which we do in our classification scheme.

The work presented by Crestan et al. has been further extended by Lautert et al. (2013), who consider two layers of classification. The first layer is similar to the classification scheme proposed before, classifying tables into one of five categories: vertical, horizontal, matrix, formatting and navigational. Again, the classification scheme includes genuine and layout tables, but does not consider lists or forms. A secondary classification scheme further classifies genuine tables based on structural characteristics, such as the occurrence of merged cells or nested tables. This classification scheme includes the following categories: concise, nested, multivalued (simple and composed), and splitted. These classes indicate the occurrence of special features in the table structure and can be used to direct subsequent table analysis and interpretation tasks. For this review, we focus on the primary classification based on table layouts. The authors consider a set of 25 features, with 20 features adopted from Crestan et al. and 5 features added to address multivalued tables. A neural network is used to classify the Web tables. A set of $4,000$ manually labeled tables are evaluated via 10-fold cross-validation and the results compared to the work of Crestan et al. The authors note an increase in classification performance for all categories, except for matrix tables. The weak performance for matrix tables is attributed to the fact that too few matrix tables are included in the corpus to allow for sufficient training. A significant increase in classification accuracy is noted for vertical tables (which correspond to horizontal listings in the classification scheme of Crestan et al.). However, this result is mainly attributed to the higher occurrence of these tables in the test corpus and the higher quality of tables, in general, as many tables were extracted from Wikipedia, which has a high table quality compared to the overall Web.

Both table layout classification approaches are very similar and incorporate the detection of genuine relational tables into the classification of table layouts. The classifiers achieve good results for both classification aspects. However, no comparison to a two-stage classification approach that performs each task separately is provided.



(a) Single Consolidated Task (b) Separate Tasks

Figure 4.2: Alternative classification approaches. Rectangular boxes represent the valid classes considered for each classification problem. Highlighted in blue are the layout classes we are interested in.

4.2 CLASSIFICATION PROBLEM

After reviewing previous work related to Web table detection and layout classification, we now take a closer look at the specific classification approaches we wish to compare. First, we recall the tasks we want to carry out as part of the classification process: (1) the detection of *genuine* relational tables, and (2) the identification of the layout type of genuine tables.

Genuine Tables
The objective of table detection is to identify tables that represent actual relational tables, i.e. they match the characteristics of tables as defined in Chapter 2. These tables frequently contain simple strings or numeric values in the table cells. Moreover, they feature syntactic similarities between values belonging to the same attribute domain, reflected by coherent rows or columns. In the detection process, candidate tables are classified as either *genuine* or *non-genuine*. Consequently, table detection can be regarded as a binary classification problem. In the context of Web pages, non-genuine tables are mostly HTML tables used for layout purposes or to represent menu structures.

Table Layout
The objective of layout type identification is the analysis of a table's logical structure. Although table structures on the Web are very heterogeneous, several prominent structures can be identified, representing the main layout types. In this thesis, we consider three main layout types: horizontal listings, vertical listings and matrix tables. Most genuine tables can be assigned to one of these layout types. Therefore, layout identification is an n-ary classification problem, where n is the number of specified layout types. The layout types we consider are based on the alignment of values of the same attribute within a table, as described in Section 3.1. Consequently, characteristic features to identify each type are the location of attribute labels and the coherence of values per row or column.

Classification Methodology
To carry out both of these tasks, we consider two different classification approaches, a single-layer and a double-layer approach. The single-layer approach combines both classification problems in a

single classification task. As illustrated in Figure 4.2(a), non-genuine or layout tables are regarded as one class in the classification scheme, similar to the layout classes *Vertical*, *Horizontal* and *Matrix*. An additional class *Other* is included for all tables that do not fit into any of the previous classes. The same features, classifier and training data are used to classify all Web tables in this approach. Consequently, selected features must be suitable for table detection as well as layout identification.
In contrast, a double-layer approach performs two separate classification tasks consecutively, using the output of the table detection task as the basis for table layout identification. As only the layout type of genuine tables is of interest, non-genuine tables are discarded after the first classification step, as illustrated in Figure 4.2(b). As both tasks are performed independently, different features, classifiers and training data can be applied.
Our objective is to identify which approach is better suited and achieves the best overall accuracy for the detection and analysis of Web tables. We consider a wide range of characteristic features and various classification algorithms to provide a comprehensive comparison.

4.3 FEATURE SPECIFICATION

In the literature, a wide range of features have been proposed for the identification of genuine Web tables and the classification of table layouts. We consider features at two different levels of granularity: global features and local features. We take into account structural as well as content features, utilizing the presence of HTML markup if applicable. The features address different aspects of table detection, layout classification or both. We do not specifically distinguish between features suitable for table detection and layout classification, respectively, as one of the approaches we intent to evaluate incorporates both tasks into a single classification problem.

4.3.1 Table Features

Global features describe the table as a whole and, thus, are computed once per table. As global features, we take into account the general table structure of rows and columns, the overall consistency of cell entries, the distribution of different data types, as well as the occurrence of designated header tags. These features incorporate and extend the features proposed by (Crestan et al., 2011) as well as (Y. Wang et al., 2002).

Table Structure
Table structure features describe the size and orientation of a table. They take into account the extent of rows, column and cells. As global features, we consider the maximal extent as well as the average extent across the table, as defined in Table 4.1. These features provide a first indication whether a candidate table has the regular structure that is common for genuine tables. Furthermore, they detect very small tables that are unlikely to represent relational content.

Consistency and Variation
In addition to the previous features, which describe the general extent of a table, we also consider the variation encountered in the extent of different table segments. From this variation, we can derive a more precise measure of the regularity of the table structure. In particular, we consider the standard

Table 4.1: Global features describing the table structure.

Feature	Description
• MAX_ROWS	Maximal number of cells per row, which are not created by a <SPAN> tag.
• MAX_COLS	Maximal number of cells per column, which are not created by a <SPAN> tag.
• MAX_CELL_LENGTH	Maximal number of characters per cell.
• AVG_ROWS	Number of cells per row, averaged across all rows.
• AVG_COLS	Number of cells per column, averaged across all columns.
• AVG_CELL_LENGTH	Average number of characters per cell.

Table 4.2: Global features measuring variation in the table structure.

Feature	Description
• STD_DEV_ROWS	Standard deviation of the number of cells per row.
• STD_DEV_COLS	Standard deviation of the number of cells per column.
• STD_DEV_CELL_LENGTH	Standard deviation of the number of characters per cell.

deviation of the size of rows, columns and cells (see Table 4.2). The standard deviation of a measure s for a set of n instances is defined as follows:

$$\texttt{STD\_DEV} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(s_i - s_{avg})^2} \tag{4.1}$$

In addition to the variance in the table structure, we consider the consistency of the table entries with respect to their size. For each cell c, we consider the size s as the number of characters and compute the *cumulative length consistency (CLC)* per row or column as follows, where s_{avg_i} is the average cell size of table segment (i.e. column or row) i:

$$CLC_i = \sum_c 0.5 - x_i, \text{where } x_i = \min\left(\frac{|s_c - s_{avg_i}|}{s_{avg_i}}, 1\right) \tag{4.2}$$

These consistency scores are averaged across all rows (CLC_R) and columns (CLC_C) and the maximum is returned as the global length consistency.

$$\texttt{CUMULATIVE\_LENGTH\_CONSISTENCY} = \max\left(CLC_R, CLC_C\right) \tag{4.3}$$

Table 4.3: Global features measuring content ratio.

Feature	Description
• RATIO_IMG	Cells containing <IMG> tag.
• RATIO_FORM	Cells containing <FORM> tag.
• RATIO_HYPERLINK	Cells containing <A> tag.
• RATIO_ALPHABETIC	Cells with predominantly alphabetic characters.
• RATIO_DIGIT	Cells with predominantly numeric characters.
• RATIO_EMPTY	Empty cells.
• RATIO_OTHER	Cells not matching above categories.

Content Ratio

The content ratio features identify what kind of content or data type is predominant in a table. Layout tables often contain many images or hyperlinks, while relational tables rather contain simple data types such as character strings or numeric values. We consider five different content types: images, forms, hyperlinks, alphabetic characters and numeric characters. In addition, we look for empty cells. An additional category *Other* is added to account for cell entries that do not match any of the previous categories. Table 4.3 lists all global content ratio features.

The ratio of cells containing content of a specific type t is then defined as follows, where n is the number of cells in a table and t_i is the content type of cell i.

$$\texttt{RATIO} = \frac{1}{n}\sum_{i=1}^{n} x_i, \text{ where } x_i = \begin{cases} 1, & \text{if } t_i = t \\ 0, & \text{else} \end{cases} \tag{4.4}$$

In addition to the content ratio, we also consider the general content type consistency. The consistency is first analyzed per row or column and then averaged across the table. For each table segment, i.e. row or column, we compute the *cumulative type consistency (CTC)* as follows, where dt_i is the dominant content type in segment S_i:

$$CTC_i = \sum_{c \in S_i} x_c, \text{ where } x_c = \begin{cases} 1, & \text{if } t_c = dt_i \\ -1, & \text{else} \end{cases} \tag{4.5}$$

The consistency scores are then averaged across all rows (CTC_R) and columns (CTC_C). As the global content consistency feature, we take the maximum of these scores.

$$\texttt{CUMULATIVE\_CONTENT\_CONSISTENCY} = \max\left(CTC_R, CTC_C\right) \tag{4.6}$$

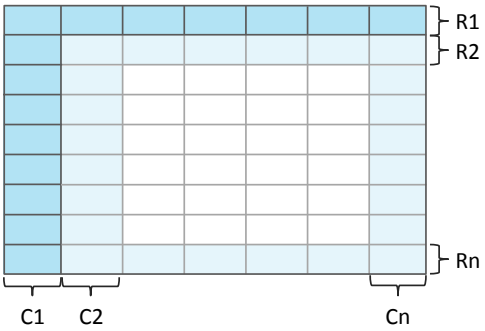


Figure 4.3: Local features are computed for the first two rows and columns as well as the last row and column of a table.

Header
In HTML tables, the designated `<TH>` tag can be used to define header cells. Layout tables generally do not contain a header. Therefore, the presence of markup for header cells is a good indicator for genuine tables. The corresponding feature is defined as follows:

$$\texttt{HAS\_HEADER} = \begin{cases} 1, & \text{if table contains header markup} \\ 0, & \text{else} \end{cases} \tag{4.7}$$

4.3.2 Subset Features

In addition to global features, we consider a number of local features, which are computed for subsets of the table. We follow the approach proposed by Crestan et al. (2011) and consider only the first two rows, the first two columns, the last row and the last column of a table as segments for local features (see Figure 4.3). The first row and the first column of a table are potential locations for the label cells. Computing features for these segments and comparing them to their neighboring segments enables recognizing the orientation of headers in the tables, which is an important indicator for the layout type. Furthermore, considering the cell content at the beginning and end of each column or row provides an estimate of the coherence of the content as well as the orientation of the table. As local features, we again consider structural features as well as the content ratio.

Structural Features
As structural features of a table segment S_i, we take into account the average and variance of the size s of cells in the segment. Each of these features is computed once for each of the selected segments.

$$\texttt{LOCAL\_LENGTH\_AVG} = \frac{1}{|S_i|} \sum_{c \in S_i} s_c \tag{4.8}$$

$$\texttt{LOCAL\_LENGTH\_VARIANCE} = \frac{1}{|S_i|} \sum_{c \in S_i} (s_c - s_{avg_i})^2 \tag{4.9}$$

Table 4.4: Local features measuring content ratio.

Feature	Description
• LOCAL_RATIO_HEADER	Cells containing <TH> tag.
• LOCAL_RATIO_ANCHOR	Cells containing <A> tag.
• LOCAL_RATIO_IMAGE	Cells containing <IMG> tag.
• LOCAL_RATIO_INPUT	Cells containing <INPUT> tag.
• LOCAL_RATIO_SELECT	Cells containing <SELECT> tag.
• LOCAL_RATIO_FONT	Cells containing <B>, <I>, <U> or <FONT> tags.
• LOCAL_RATIO_BR	Cells containing tag.
• LOCAL_RATIO_COLON	Cells containing colons.
• LOCAL_RATIO_CONTAINS_NUMBER	Cells containing numeric characters.
• LOCAL_RATIO_IS_NUMBER	Cells containing *only* numeric characters.
• LOCAL_RATIO_NON_EMPTY	Cells that are not empty.
• LOCAL_RATIO_UNORDERED_LIST	Cells containing <UL> tag.
• LOCAL_RATIO_ORDERED_LIST	Cells containing <OL> tag.
• LOCAL_RATIO_COMMA	Cells containing commas.
• LOCAL_RATIO_BRACKET	Cells containing brackets "(" or ")".

Additionally, we consider the distribution of virtual cells in the segments. Virtual cells are created within a table cell via <SPAN> tags. A single table cell can contain multiple <SPAN> tags. We compute the local span ratio as another feature for each table segment, considering all physical and virtual cells c in a segment.

$$\texttt{LOCAL\_SPAN\_RATIO} = \frac{1}{|S_i|} \sum_{c \in S_i} x_c, \text{ where } x_c = \begin{cases} 1, & \text{if } c \text{ is a virtual cell created by a <SPAN> tag} \\ 0, & \text{else} \end{cases} \tag{4.10}$$

Content Ratio

Similar to the global content ratio features, we also consider the content type of cells in each local segment. We consider a wide range of content types, based on the presence of special tags or characters. The ratio is computed as before. However, in contrast to the global features, the content types considered here are not mutually exclusive. Multiple types can be assigned to the content of a single cell. Table 4.4 describes the content types and corresponding features. Images, hyperlinks or forms are not frequently found in genuine relational Web tables and, consequently, indicate that a table is used for layout purposes. Lists, commas or line breaks suggest that the content of a cell is not atomic. Yet, well-formed relational tables predominantly contain atomic values. Other tags, such as text formatting tags, can be used to estimate the coherence and orientation of a table, as attribute values from the same domain generally feature similar formats.

4.3.3 Pre-Selection Filters

Similar to previous Web table detection approaches, we can eliminate candidate tables, where it is obvious that they do not represent genuine tables, using a set of simple rules (see Section 4.1 for more details). Similar to the rules proposed by Penn et al. (2001), we expect genuine tables to feature *at least* two rows and two columns. Otherwise, the table resembles a list or single cell, which we do not consider a genuine table.
Additionally, we remove tables that are invalid, i.e. the table structure does not form a valid HTML segment, as well as tables that cannot be displayed correctly, which indicates a low quality of the table in general. Surprisingly, the majority of HTML tables on the Web does not pass these simple filters, which amplifies the importance of accurate table detection algorithms to identify relevant, high quality tables amongst all these potential candidates.

4.4 EXPERIMENTAL EVALUATION

In order to establish, which of the two classification approaches we described in Section 4.2 achieves a higher overall accuracy for Web table detection and layout classification, we evaluate both approaches on a corpus of tables extracted from the Web. We compare the results of different classification algorithms to ensure that any difference in performance between the two approaches is not simply due to the suitability of the selected algorithm. Additionally, we consider feature selection to reduce the dimensionality of the problem and ensure that we only use effective features for each classification task. The feature selection also provides insights into the different characteristics of table detection and layout identification.

4.4.1 Dataset

For the test corpus, we utilize the *Common Crawl*[1], a publicly available archive of crawled Web pages, published by the Common Crawl Foundation. The archive is regularly extended with new Web crawls. For our evaluation, we used the crawl published in October 2014. From a subset of the crawl, we randomly extracted $26,654$ HTML tables. From these tables, we identified $24,623$ tables as obviously non-relational using the simple filter rules described in Section 4.3.3. After this initial filter step, $2,022$ tables remained in the corpus, which were manually labeled and used for evaluation. Table 4.5 shows the frequency with which each table class occurs in the corpus. In total, layout tables make up about 96% of all tables that we extracted from the Web. This percentage initially seems very high, but matches the estimate of 98.9% reported by Cafarella et al. (2008b) for a large corpus of 14.1 billion HTML tables extracted from the Web (although only vertical listings are considered as genuine tables, which results in a slightly higher ratio of non-relational tables). Furthermore, the distribution of table layout types for genuine tables also matches the relative distribution of these layout types on the Web (see Appendix B for an estimate). Since matrix tables are significantly less frequent on the Web than tables with other layouts, our test set contains only a few instances of matrix tables. The small sample size for this class can potentially impact the performance of any classification algorithm. A similar issue has been reported by Lautert et al. (2013).

[1] `https://commoncrawl.org`

Table 4.5: Distribution of table and layout classes in the test corpus.

Class	# of Tables
Layout	1,092
Genuine	930
Vertical Listings	316
Horizontal Listings	434
Matrix	35
Others	145

4.4.2 Feature Selection

In Section 4.3, we specified a large set of features for our classification problems, leading to a high-dimensional feature space. In total, we consider 127 features per table. Depending on the classification algorithm used, a large number of features often requires more training data to achieve good prediction results and the separation of classes can be more challenging in high-dimensional spaces. To reduce the dimensionality and to ensure that no redundant features are considered, we first perform feature selection.

We employ *Correlation-based Feature Selection (CFS)*, developed by M. A. Hall (1999), which is a filter approach that is independent of any specific classification algorithm. CFS recursively selects features that increase the so-called *merit* of the feature set, until no additional feature adds any benefit. To specify the merit of the feature set, CFS uses Pearson's correlation coefficient, which is biased towards features that are highly correlated with a class variable, but uncorrelated with other features in the set. As a result, irrelevant and redundant features are removed from the original feature set without loss of classification accuracy (M. A. Hall, 1999).

We use the implementation of CFS available in the WEKA machine learning toolkit (M. Hall et al., 2009) and apply it to each classification problem individually, i.e. table detection and layout classification, as well as to the consolidated classification task. The selection will indicate if any features are only relevant for one of the tasks.

From the initial set of 127 features, the CFS algorithm reduced the number of features to 29 for the table detection problem, to 23 for the layout identification problem and to 31 for the combined classification task. This means a significant reduction in dimensionality for each of the classification problems. Table 4.6 shows the features selected for each task, with content ratio features combined. Four features, namely `MAX_ROWS`, `MAX_CELL_LENGTH`, `STD_DEV_CELL_LENGTH` and `CUMULATIVE_LENGTH_CONSISTENCY` are not selected for any of the classification problems, most likely because they are correlated with other features. There are apparent differences in the feature sets selected for each problem. While, for instance, the presence of a header or tags is useful to detect genuine tables, the average cell size as well as column and row sizes are more relevant for the identification of a table's layout. Similarly, content ratio features selected for each classification problem differ, as well. While the ratio of cells containing images or forms is relevant for table detection, it is not relevant for layout identification. Instead, the local ratio of header cells and cells containing

Table 4.6: Features selected for each classification task.

Feature	Table Detection	Layout Identification	Combined
MAX_COLS		×	×
AVG_COLS		×	×
AVG_ROWS	×		
AVG_CELL_LENGTH		×	
STD_DEV_COLS		×	
STD_DEV_ROWS		×	
RATIO_X	×	×	×
CUMULATIVE_CONTENT_CONSISTENCY	×	×	×
HAS_HEADER	×		
LOCAL_LENGTH_AVG	×	×	×
LOCAL_LENGTH_VARIANCE	×	×	×
LOCAL_SPAN_RATIO	×		×
LOCAL_RATIO_X	×	×	×

colons, which also indicates label cells in some tables, are selected. The reduced feature sets clearly reflect the different objectives of the classification problems. Consequently, the features selected for the combined classification task appear to be a combination of features selected for the individual tasks. The different feature sets selected by CFS suggest that a double-layer classification approach has the potential to outperform a single-layer approach, by tailoring each classification step to the characteristics of the classification problem at hand. In the next section, we evaluate each approach using different classification algorithms to see if we can confirm this hypothesis.

4.4.3 Classifiers

In our evaluation, we consider various classification algorithms, most of which have been successfully applied to similar tasks in the literature. We consider two classes of algorithms, decision trees and support vector machines. Previous classification results reported in the literature indicate that the choice of classifier has significantly less impact on the classification performance than the selection of appropriate features. However, we consider several classification algorithms to ensure that a better performance of one of the classification approaches (i.e. single-layer or double-layer approach) is not simply due to the performance of the classification algorithm. For the classification and evaluation, we, again, utilize the WEKA machine learning toolkit.

Decision Trees

As the first class of classification algorithms, we consider *decision trees*, which have been successfully applied to similar tasks by Y. Wang et al. (2002) as well as Crestan et al. (2011). Decision tree algorithms, as implied by the name, use a tree structure as the classification model. Starting at the root,

the dataset is split at each node, by evaluating the values of a single selected feature. The procedure is repeated for each child node, until one of the leaves, which represent the class labels, is reached. During the training phase, a split criterion determines for each node, which feature is evaluated, and stop criteria terminate the recursive procedure. Decision trees are frequently used, as they are robust, easy to interpret and handle heterogeneous features without making any assumptions about the independence of the features. In detail, we consider the following decision tree algorithms:

- **CART:** Classification and Regression Trees (CART) is a simple, non-parametric decision tree algorithm. (Breiman et al., 1984) While learning the tree, at each node the algorithm selects the feature that best splits the classes in the dataset. As the split criterion, CART uses different measures, including the Gini impurity. The same split procedure is repeated for each child node until a stop criterion is met. For our evaluation, we use *SimpleCART*, an implementation of the decision tree algorithm supplied by WEKA.

- **C4.5:** C4.5 is another popular decision tree algorithm, developed by Quinlan (1993). The algorithm is very similar to CART, but uses an information gain ratio as the split criterion at each node. In addition, C4.5 applies pruning to reduce the tree size and, thus, avoid overfitting. We use the WEKA implementation of the algorithm named *J48*.

- **Random Forest:** As an alternative to simple decision trees, we also consider the random forest algorithm, developed by Breiman (2001), which represents an ensemble of decision trees. Using bagging, the training data is sampled to generate multiple different decision trees and the individual classification results are aggregated to produce the final prediction. By using multiple decision trees generated from samples of the data, random forest avoids overfitting, which is a common issue for decision tree algorithms.

Support Vector Machines

As a second class of classifiers, we consider *support vector machines (SVM)*, which were first developed by Vapnik (1982). In their original form, SVMs represent non-probabilistic linear binary classifiers. Considering each data instance as a point in high-dimensional feature space, the SVM algorithm defines a hyperplane that separates the classes, maximizing the distance of instances of each class to the hyperplane. Unseen instances are then classified based on which side of the hyperplane they are located in feature space. While a hyperplane can only define a linear segmentation, introducing a kernel function allows for a non-linear segmentation (Boser et al., 1992). Popular kernel functions are the polynomial kernel and the radial base function (RBF) kernel.
SVMs are originally binary classifiers, which means that they can only distinguish between two classes. However, SVMs can be extended to support multiple classes, for example by regarding them as multiple binary classifications.
SVMs have been used before by Y. Wang et al. (2002) as well as Son et al. (2013) to detect genuine HTML tables. In our experiments, we use an implementation of support vector machines provided in WEKA named *SMO*, which uses the sequential minimal optimization algorithm developed by Platt (1998) to train the classifier. We consider both, a polynomial kernel and an RBF kernel.

4.4.4 Evaluation

To provide a comprehensive comparison and evaluation of the different processing approaches for table detection and layout classification, we conduct a range of experiments. We evaluate the classification performance using repeated random subsampling. We randomly split the dataset, using 90% of the data for training and 10% for validation. All results are averaged over 100 iterations.

Metrics
A common metric for classification performance is *accuracy*, which measures the number of correct predictions divided by the number of all predictions. However, especially for unbalanced datasets, where one category is predominant, accuracy is often not sufficient to evaluate the prediction quality of a model. As our goal is to achieve a high classification performance for the less frequently represented classes *Vertical Listing*, *Horizontal Listing* and *Matrix*, we require more suitable metrics. Therefore, we use the following metrics to evaluate classification performance:

- **Precision:** For each class, the precision measures the number of correct predictions divided by all predictions for this class. The overall precision is then computed as the (weighted) average of the precision values per class.

$$Precision = \frac{TP}{TP + FP} \tag{4.11}$$

- **Recall:** Recall for each class is calculated as the number of correctly predicted instances divided by the number of all instances belonging to a class. Again, the overall recall is the (weighted) average of recall values per class.

$$Recall = \frac{TP}{TP + FN} \tag{4.12}$$

- **F-Measure:** The F_β-measure combines precision and recall into a single score, using a parameter β to control whether more emphasis is put on precision or recall. The most commonly used measure is F_1, which balances the impact of precision and recall.

$$F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \tag{4.13}$$

Classification Problems
In the first set of experiments, we study the suitability of different classification algorithms with respect to the individual classification problems. Therefore, we evaluate the table detection and layout identification tasks individually as well as combined in a single classification task. For each classification problem, we use the full set of features and measure precision, recall and F_1 per class. Additionally, we measure the weighted average for each metric, using the class frequencies as weights to account for the unbalanced distribution of classes in the corpus.
First, we evaluate the table detection task, where we distinguish between layout tables and genuine tables. The performance measures are presented in Table 4.7. Overall, the results show a very similar performance for all tested classification algorithms, with the Random Forest classifier performing best

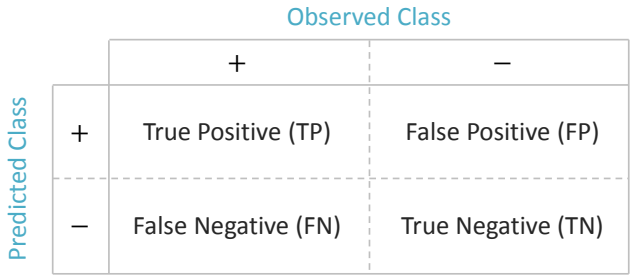


Figure 4.4: Confusion matrix specifying the terms used in Equations 4.11 and 4.12, distinguishing between instances that belong to a specific class (+) and those that do not belong to the class (-). The observed class is the true class of an instance, whereas the predicted class is the class assigned by the classifier.

with respect to F_1. For all classifiers, the prediction quality for layout tables is much better regarding precision and recall than for genuine tables, which is a very heterogeneous class due to the different layout types. However, we achieve a significantly higher precision for genuine tables compared to 41% precision achieved by the approach proposed by Cafarella et al. (2008a), which is frequently used by other researchers.

Next, we study the quality of the layout identification task. For the evaluation, we only consider genuine tables in the training and test sets. All layout tables are removed from the corpus, and we only consider classes *Vertical Listing*, *Horizontal Listing*, *Matrix* and *Other*. The results are shown in Table 4.8 and confirm that layout classification is more challenging compared to table detection. Overall, we observe significantly more variation between the different classes and also slightly more variation between the different classifiers. Again, decision tree classifiers perform slightly better overall, with the best results achieved by SimpleCART and Random Forest. At class level, we achieve good results for vertical and horizontal listings, yet achieve only low precision and recall for matrix tables. This issue, which has also been reported by Lautert et al. (2013), is mainly due to low number of matrix tables in the dataset. As a result, there are not enough training samples for this class to build a reliable model and make accurate predictions. This is confirmed by the results achieved by the support vector machine with a polynomial kernel, which are slightly better for matrix tables compared to other classifiers. SVMs are known to generalize well even for a small number of training samples. However, overall a larger training set is necessary to improve the prediction performance for matrix tables.

Finally, we measure the performance of the combined classification task. The valid classes in this experiment correspond to the classification scheme in Figure 4.2(a). The results for each classification algorithm are included in Table 4.9. Again, we observe only little variation between the different classification algorithms, with Random Forest performing best. The prediction quality per class is similar to the results reported by Lautert et al. (2013). The prediction of layout tables is very accurate, while we observe a lower precision and recall for horizontal and vertical listings. Again, we experience issues with matrix tables, due to their low frequency in the dataset.

So far we have utilized the complete set of features for all classification problems. In the next set of experiments, we evaluate the impact of feature selection for each task.

Table 4.7: Evaluation of the table detection task. All measures are reported for the following classes: Layout (L) and Genuine Tables(G).

Classifier	Metric	L	G	Weight. Avg.
J48	Precision	95.11	82.64	89.38
	Recall	94.94	83.17	89.53
	F1	95.02	82.82	89.41
SimpleCART	Precision	95.17	81.61	88.93
	Recall	94.55	83.38	89.41
	F1	94.85	82.39	89.12
Random Forest	Precision	94.92	**87.30**	**91.42**
	Recall	**96.53**	82.12	89.90
	F1	**95.71**	**84.55**	**90.58**
SMO (Poly)	Precision	94.74	81.99	88.87
	Recall	94.79	81.81	88.82
	F1	94.75	81.81	88.80
SMO (RBF)	Precision	**95.43**	83.58	89.98
	Recall	95.19	**84.23**	**90.15**
	F1	95.30	83.81	90.02

Feature Selection

As detailed in Section 4.4.2, we applied correlation-based feature selection (CFS) to identify the most relevant features for each classification problem and reduce the dimensionality by removing irrelevant or redundant features. We repeated all previous experiments for all classifiers, using only the selected features. In Table 4.10, we compare the weighted average F_1 measures for each experiment to evaluate the impact of the feature selection. In most cases, the performance measures achieved with only the selected features are very similar to the values achieved with all features. This confirms that the selection algorithm was successful in removing irrelevant and redundant features. Only the performance of support vector machines declined noticeably, especially in the layout classification experiment. We attribute these changes mainly to the selection of kernel parameters. Due to the reduced dimensionality, the optimal kernel parameters most likely differ from the previous settings, and a more comprehensive parameter optimization is required to achieve the best results. Apart from this, the dimensionality reduction has only little impact on the prediction performance. However, feature selection also reduces the computational costs for the classification tasks, which is especially relevant for the processing of huge Web corpora.

Single-Layer vs. Double-Layer Classification

After evaluating all individual classification tasks as well as the impact of feature selection, we now analyze the results of the main experiment, which compares the single-layer and the double-layer classification approaches, as described in Section 4.2. The single-layer approach corresponds to the combined classification task in the previous experiments. For the double-layer approach, we perform table detection and layout identification consecutively, using the tables classified as *genuine* in the

Table 4.8: Separate evaluation of the layout identification task. All measures are reported for the following classes: Vertical Listings (V), Horizontal Listings (H), Matrix (M) and Other (O).

Classifier	Metric	V	H	M	O	Weight. Avg.
J48	Precision	**73.62**	88.48	31.28	66.39	77.83
	Recall	78.30	88.17	24.08	**61.13**	78.19
	F1	75.50	88.17	24.63	62.95	77.54
SimpleCART	Precision	73.08	**90.51**	30.78	76.70	**80.18**
	Recall	**88.83**	88.20	15.37	57.12	**80.83**
	F1	**79.85**	89.17	18.84	64.01	**79.43**
Random Forest	Precision	71.22	90.02	35.70	**80.98**	**80.18**
	Recall	86.87	89.24	17.93	56.90	80.71
	F1	77.98	**89.50**	21.69	**65.87**	79.35
SMO (Poly)	Precision	71.21	85.68	**49.67**	65.87	76.32
	Recall	79.04	84.59	**37.36**	56.89	76.61
	F1	74.57	85.01	**39.41**	60.12	75.86
SMO (RBF)	Precision	72.16	85.22	26.33	79.34	77.65
	Recall	85.09	**89.34**	7.84	54.56	79.40
	F1	77.81	87.09	11.51	63.62	77.43

Table 4.9: Evaluation of combined classification problem. All measures are reported for the following classes: Layout (L), Vertical Listings (V), Horizontal Listings (H), Matrix (M) and Other (O).

Classifier	Metric	L	V	H	M	O	Weight. Avg.
J48	Precision	**94.24**	63.08	73.45	19.89	58.46	81.05
	Recall	94.63	66.33	**72.63**	16.35	49.48	80.89
	F1	94.42	64.26	72.80	16.31	51.97	80.67
SimpleCART	Precision	92.73	64.74	76.81	1.50	64.03	81.30
	Recall	95.45	**69.15**	68.00	1.33	45.33	80.22
	F1	94.05	66.46	71.62	1.39	51.94	80.30
Random Forest	Precision	93.12	67.51	**84.78**	35.83	**76.39**	**85.13**
	Recall	**97.34**	67.28	71.03	14.65	48.50	**82.06**
	F1	**95.17**	67.09	**77.09**	19.31	58.40	**82.95**
SMO (Poly)	Precision	94.15	66.72	73.00	**40.86**	61.08	82.03
	Recall	95.62	67.59	70.21	**33.12**	47.47	81.24
	F1	94.87	66.73	71.23	**32.52**	52.43	81.28
SMO (RBF)	Precision	92.83	**69.71**	78.68	14.00	76.24	83.62
	Recall	96.97	68.95	67.31	3.57	**50.20**	81.25
	F1	94.84	**68.90**	72.31	5.61	**59.63**	81.88

Table 4.10: Weighted F_1-measure achieved using all features compared to features selected by CFS algorithm.

Classifier	Table Detection		Layout Identification		Combined	
	All	CFS	All	CFS	All	CFS
J48	89.41	89.22	77.54	78.46	80.67	80.69
SimpleCART	89.12	89.56	79.43	77.92	80.30	80.16
Random Forest	**90.58**	90.52	79.35	**80.56**	**82.95**	82.93
SMO (Poly)	88.80	89.08	75.86	73.42	81.28	80.79
SMO (RBF)	90.02	88.63	77.43	72.86	81.88	79.44

first step as input for the second step. We use the same training set to train both classifiers. We then evaluate the test set and measure the overall prediction quality.

As the previous results indicated no major differences between the classification algorithms, we only use the Random Forest algorithm in this experiment. Although we could combine different classifiers in the double-layer approach, Random Forest was among the best algorithms for each classification problem. Additionally, for each task, we only use the features selected by the CFS algorithm.

The results of this experiment are presented in Table 4.11. In general, we can observe that both approaches achieve very similar results, which means that there is no clear winner. For both approaches, we observe the best prediction performance for layout tables and slightly lower measures for vertical and horizontal listings. Also, both approaches show a weak performance for matrix tables. In general, the results do not confirm the initial assumption that a double-layer approach achieves more accurate results. Comparing the results to the performance measures achieved by layout identification separately, we can see that errors from the table detection task propagate through the classification process and impact the precision in the layout identification step.

Although the single-layer and double-layer approaches achieve a very similar prediction quality overall, if we look at the individual measures in detail, we can identify some differences in their respective performances. While the single-layer approach achieves a higher *precision* for genuine tables, the double-layer approach achieves a higher *recall* for most of the classes. Depending on the target application of the extracted tables, we may favor one over the other. Cafarella et al. (2008a), for instance, explicitly favor recall for the extraction of genuine tables, as the precision can be further improved in subsequent processing steps.

In addition, there are further characteristics that need to be taken into account when selecting one of the classification approaches. On the one hand, a single-layer approach is less computationally expensive, as it involves only a single classification step. On the other hand, the double-layer approach is more flexible, providing opportunities for the training data, feature and classifier selection to be adjusted to the task at hand.

Table 4.11: Comparison of the single-layer and double-layer approaches, using Random Forest as the classification algorithm and the features selected by CFS.

Approach	Metric	L	V	H	M	O	Weight. Avg.
Single-layer	Precision	93.57	**66.27**	**81.32**	**39.91**	**76.65**	**84.45**
	Recall	**96.60**	69.60	71.95	**17.83**	50.83	82.44
	F1	95.06	67.50	76.10	**22.67**	**60.14**	82.93
Double-layer	Precision	**95.29**	64.35	78.51	26.63	68.15	83.72
	Recall	96.16	**73.26**	**73.93**	15.86	**53.38**	**83.35**
	F1	**95.72**	**68.52**	**76.15**	19.88	59.87	**83.38**

4.5 SUMMARY AND DISCUSSION

In this chapter, we studied two aspects of Web table recovery, the detection of genuine tables as well as the identification of layout types. Accurate table detection is essential for the extraction of table data from the Web, as the vast majority of table structures in HTML pages are used for layout purposes, and only a small share contains the kind of relational information we are interested in. These *genuine* Web tables, however, are not uniformly structured. Instead, different layout types are used to represent the data, depending on the content and purpose of the table.
Layout classification allows us to identify the main layout categories of genuine Web tables. Based on these categories, we can then make more accurate assumptions about table characteristics, such as the location of the header, and, ultimately, the semantics of the table. However, in the literature, table layout classification has received only little attention, whereas Web table detection has been studied in greater detail. Instead of distinguishing between different layout types, many table analysis and interpretation techniques simply assume a uniform layout across all tables. In many cases, a simple layout similar to the layout commonly used for database tables, with attribute labels at the top of each column and no designated stub, is expected. It is clear that assuming a single uniform layout either excludes a substantial number of tables from the recovery process or leads to inaccurate results.
Therefore, in this chapter, we focused on incorporating layout classification in the Web table recovery process. In particular, we studied two alternative approaches. The first, which has been proposed in the literature, combines layout classification with table detection into a single classification task. The second treats both problems as separate consecutive classification tasks. We conducted a comparative evaluation on real-world data to establish, which approach is more effective. In detail, we addressed the following aspects:

- **Classification Scheme:** Incorporating lessons learned from related work, we proposed a classification scheme for genuine Web tables, that distinguishes three main layout types: vertical listings, horizontal listings, and matrix tables.

- **Feature Selection:** We consolidated and extended a wide range of features proposed in the literature for each of the classification tasks. Using correlation-based feature selection, we evaluated the relevance of each feature with respect to the classification problems. The selected

feature sets offer a first indication regarding the suitability of the single-layer and double-layer approach, respectively.

- **Experimental Evaluation:** We conducted an experimental evaluation on a corpus of real tables extracted from the Web. We evaluated both approaches, comparing various classification algorithms to ensure that the results are not influenced by the characteristics of a single algorithm. The results of our experiments indicate that both approaches are equally suited to incorporate layout identification for Web tables, although a double-layer approach provides more flexibility to adjust to the data as well as the specific classification tasks.

Our comprehensive experiments also confirmed an issues that has previously been reported in the literature. The different layout types observed in Web tables are not equally frequent on the Web. Especially matrix tables are relatively rare. Consequently, a large sample size is required, when the training data is randomly sampled from the Web, in order to include a sufficient number of matrix tables to build the classification model. Additionally, layout tables represent the vast majority of tables on the Web. Thus, the accuracy with which these tables are identified and filtered, has a significant impact on the precision of detecting genuine tables with different layout types.
Overall, we achieve good results for the detection of genuine Web tables and the identification of their main layout types. As a result, the double-layer classification approach we proposed, has been employed to classify millions of tables in the Dresden Web Table Corpus.

5

RECOVERING WEB TABLE CONTEXT

5.1 Web Table Context

5.2 Related Work

5.3 Context Relevance

5.4 Attribute-specific Context Annotation

5.5 Experimental Evaluation

5.6 Summary and Discussion

Tables form a compact representation of relational data. However, in a document, they are only one of many components that bear information (Hurst, 2000). Therefore, understanding the content and intention of a table depends on the additional contextual information contained in the document. On the Web, we encounter context in the form of headlines, captions or surrounding text. Text referring to a table can provide a summary of the content or conclusions drawn from it. It also frequently offers a more detailed description of various table entries to clarify terms or indicate restrictions on attributes (Hurst, 2000). Additionally, *hidden attributes* that have been factored out of the table are generally placed in the context (see Section 3.1). The importance of considering such context in the table recovery process is well established. A wide range of tasks, including table search (Sarawagi et al., 2014) and finding related tables (Yakout et al., 2012), benefits from this supplementary information.

However, not all information mentioned in potential context sections is actually relevant to the table. For instance, a query term that appears in the context of a table does not guarantee that the answer to the query is contained in the table. The verbosity of the context, especially when considering large texts, often introduces noise that leads to incorrect interpretations (Pimplikar et al., 2012). Consequently, evaluating the relevance of potential context segments as well as establishing an explicit link to the table content is essential in order to reduce noise and prevent misinterpretations.

In this chapter, we take a closer look at the context available for tables on the Web in order to improve its impact on Web table understanding. As part of this work, we first focus on the relevance of different context sources with respect to providing useful additional information on table content. Our objective is to identify measures that enable the evaluation of context information regarding its connection to table content and the reduction of noise based on these measures. By reducing the noise, we expect decisions based on table context as well as information extracted from it to be less ambiguous. In the second half of this chapter, we focus on utilizing the information provided in the context segments to improve the understanding of individual attributes in the table. Instead of using the context as a general descriptor of the overall topic of a table, we are interested in phrases that augment the original attribute label. For instance, the phrases "*developing country*" instead of "*country*" or "*musician*" instead of "*name*" provide a more detailed description of an attribute. Our goal is to extract these descriptive phrases, in order to provide a better table description for table matching or table search application.

In detail, this chapter is organized as follows: First, we determine the various formats and structures in which contextual information for Web tables is communicated. We also point out significant difference between embedded and external tables (Section 5.1). After that, we review related work that studies contextual information in connection with Web tables (Section 5.2). Following this review, we first concentrate on context relevance. For various context sources, we study its relation and relevance to the table, as well as its tendency to introduce noise. To address the verbosity of large context segments, we also propose a filter process that reduces noise in the context (Section 5.3). In the next section, we then study the extraction of attribute-specific information from relevant context in order to recover descriptive attribute labels (Section 5.4). We evaluate the quality and effectiveness of our contributions with a set of experiments on real-world data (Section 5.5). Finally, we conclude this chapter with a summary and discussion (Section 5.6).

5.1 WEB TABLE CONTEXT

As tables on the Web are primarily intended for human consumption, they are frequently provided with supplementary information that facilitates a swift comprehension of the table content. Such contextual information, which can be very specific to the table or of a more general nature, exists in various formats. Generally, the context available for tables embedded in HTML pages differs significantly from the context provided on designated data platforms.

Embedded vs. External Tables
The context for tables embedded in HTML pages is naturally provided by the corresponding host page (see Section 3.1). It describes the general topic and often provides more detailed or complementary information on the table content. On a Web page, we can identify various sources of contextual information. In detail, we distinguish between the following potential sources:

- **Headlines:** Page and section headlines, marked by `<H1>` ... `<H6>` tags, provide a condensed description of a topic in form of a short phrase. Multiple headlines within a page frequently form a hierarchical dependency structure.
- **Text:** Free text constitutes the biggest part of the content of most Web pages. Often, the text is divided into sections and paragraphs for better human consumption.
- **Captions:** HTML tables can provide a title or description that is displayed next to the table, via the designated `<CAPTION>` tag.
- **Hyperlinks:** Web pages are linked to other related pages via hyperlinks. The anchor text as well as the related page potentially provide additional context information.
- **Metadata:** In addition to visible content, Web pages also contain metadata that is not intended for display, but targets search engines. Common meta content includes keywords or a condensed description of the content.
- **Tables:** Other tables on the same Web page often describe related content and, thus, add to the context of the table at hand.
- **Other Media:** Further information about the general topic can also be communicated to the user via other media formats, including images, audio or video.

An example illustrating various of these context sources is provided in Appendix C.1. The most relevant context sources with respect to automatic table recovery include textual descriptions such as headlines, text and captions, as well as related tables.
In contrast to embedded tables, datasets published on designated platforms, such as Open Data portals, are stored in external files. In this case, the necessary context is provided by the access page on the platform (see Section 3.1 for more details). Often, we find additional information at resource as well as dataset level. A resource represents an individual file, for instance a table stored in CSV format, while a dataset represents a group of related resources by the same publisher that describe various aspects of the same topic. In addition to links to the resource files, access pages generally contain the following context information:

- **Title:** Resources as well as datasets feature a title that provides a condensed description of the content.
- **Description:** A more extensive textual description of datasets and individual resources provides further details. Such a text is generally intended for human consumption.
- **Metadata:** Additional technical details and other specifications about resources or datasets are frequently provided in the form of key-value pairs. Common specifications include information on the publisher as well as temporal or spatial coverage. The metadata format is optimized for automated processing of the data.
- **Related Resources:** Dataset descriptions often link to multiple related resources that comprise the dataset. In addition, links to regular Web pages, which provide a more detailed textual description of the topic and the resources, can be provided.

Again, examples for the context of a dataset as well as a resource are shown in Appendix C.2.

Context Formats
In order to identify hidden attributes, units or descriptive labels in the table context, we focus on textual descriptions, excluding other media such as images. As indicated before, these textual sources vary significantly regarding structure and information content. While headlines, titles and descriptions provide *unstructured* context information, metadata is published in a *semi-structured* fashion.
The structure, but also the extent of a segment are important to consider when analyzing and evaluating context information. Larger text segments contain regular sentences and, thus, can be analyzed using established text processing techniques, such as statistical parsers. In contrast, headlines, titles or metadata often contain only short phrases or single words, which makes it more difficult to analyze these segments algorithmically. The data provided by these segments is often not sufficient to achieve reliable results, thus, introducing noise in the context analysis. Especially parsers, but also word-based similarity and relevance measures are known to suffer from lower precision if too little data is available. Consequently, taking the characteristics of different context segments into account is important when applying these techniques.

Relation to Tables
Not all information published in connection with a table is equally relevant for the understanding of its content. Especially on Web pages, where tables represent only a small part of the content, many parts of the page are often not directly related to the specific topic of the table. As a result, considering all context equally relevant frequently introduces too much noise.
Reviewing the various types of Web table context, we can distinguish between table-specific sources and sources of more general information. Considering embedded tables, only *captions* are explicitly related to tables as they are included in the table description enclosed by `<TABLE>` tags. All other sources are only implicitly related through content or position. In contrast, context on Open Data portals is provided especially to describe resources such as tables and, as a result, a more explicit relationship exists. However, we need to distinguish between resource and dataset context. While resource context is limited to a specific table, dataset context frequently describes multiple resources and, thus, is more general and likely to introduce noise.

Table-specific sources, such as captions or resource descriptions, generally explain what the focus of the table is or how the table should be read (Hurst, 2000). It is also often used to provide additional information to distinguish the table from other tables on the same Web page or in the same dataset. For other context sources, it is less clear, how and if they are related to the table in question. Establishing an explicit connection to the table is essential to reduce noise and improve the accuracy of context-based decisions.

5.2 RELATED WORK

A table in a document is generally not an independent object, but one of many components that carry information content. Table recovery research has identified the potential for other document components, such as titles or text, to affect how table content is interpreted (Embley et al., 2006). Therefore, various work related to table recognition and table understanding, as well as applications that utilize document tables, take the context of a table into account in some form.

The table retrieval system *TINTIN*, for instance, considers text that is close to or between rows of a table (Pyreddy et al., 1997). Heuristic rules are applied to detect this text, collectively categorized as *captions*, which is then used as general information to describe the content of the table as a whole. The same rules were further extended by Pinto et al. to extract tables and captions to be used as information sources in the question answering system *QuASM* (Pinto et al., 2002). In order to improve the quality of the process involved in table and context identification and extraction, Pinto et al. proposed a classification approach based on conditional random fields to replace the heuristic extraction rules (Pinto et al., 2003). As part of the classification task, they further separated context sections into either *title* or *caption*. However, similar to the previous methods, they focus solely on context that is located directly before, after or within the boundaries of the table.

In his thesis, Hurst also recognizes the importance of contextual information for the interpretation of table content as well as for the recovery of missing labels. He extends the context to also include text segments that are not co-located with the table, such as headings and the main text, and studies formats of references to tables in the text. Such references can be *explicit*, including an index for the table, as in "shown in Table 2.2", or *implicit*, without a unique string, as in "in the following table". Furthermore, he proposes to utilize the context to improve the linguistic analysis of table cell content. Since, for instance, the identification of part-of-speech (POS) tags is unreliable for short text segments commonly found in table cells, more accurate POS tags inferred from the context can be used instead (Hurst, 2000). Overall, Hurst focuses mainly on the extraction of tables and context information, but does not consider the relevance of context segments with respect to a table, or the utilization of contextual information to interpret the table content.

In addition to table recover in general, in recent years, context has also been studied in relation to Web tables in particular, focusing mostly on tables embedded in HTML pages. Contextual information is taken into account in various applications. These include the identification of a semantic relation between tables (Yakout et al., 2012), establishing the relevance of a table in response to a search query (Limaye et al., 2010; Pimplikar et al., 2012), as well as the detection of hidden attributes (Cafarella et al., 2009; Ling et al., 2013).

The selection of contextual information that is considered suitable differs significantly amongst the individual approaches. While many approaches do not define a specific selection of context and sim-

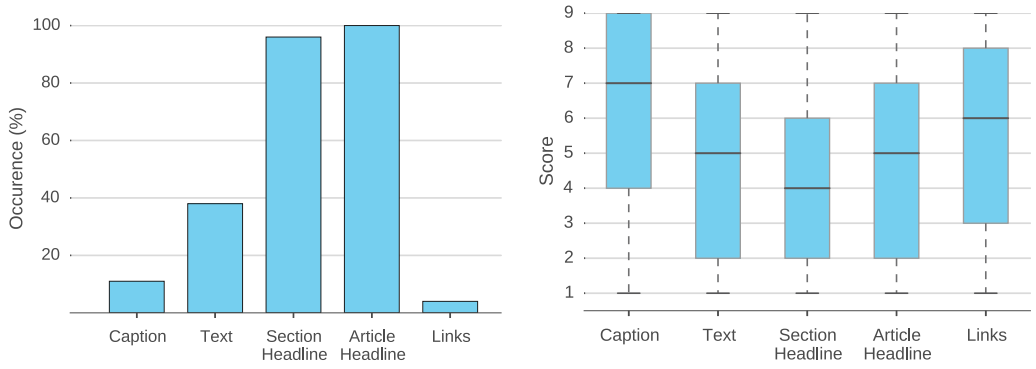
ply consider all available information, others limit the amount of information considered. Yakout et al. (2012) restrict the context to text that directly surrounds the table on the Web page. Cafarella et al. (2009) further reduce the contextual information by taking only significant terms into account, specifically the top-k terms based on *TF-IDF* scores. A more elaborate context selection technique is proposed by Pimplikar et al. (2012) and subsequently applied by Sarawagi et al. (2014). Relevant context segments are selected based on their position in the DOM tree of the document. Considered context types include heading and text segment. Starting from the path between the table node and the root of the DOM tree, all text nodes that are siblings of nodes on the path are included. In order to estimate its relevance to the table, each of these nodes is then scored based on its distance from the table, its position relative to the path as well as the occurrence of formatting tags such as *bold* or *italics*. However, as Pimplikar et al. skip further details about the extraction of context segments, the suitability of this technique is difficult to evaluate.

The selected context segments are predominantly used as an extension to the table that provides supplementary information to describe the table content. Consequently, the context is utilized to establish if a table is relevant to a search query or similar to another table. The relevance of a table in these scenarios is generally decided based on string similarity measures, with the table regarded as a bag of words. For search applications, the similarity between the search terms and the table with its context is computed, whereas integration applications measure the similarity between two tables and their respective context segments. The most frequent similarity measure is the *cosine similarity* based on *TF-IDF* scores of terms that appear in the table and context. Yakout et al. apply this measure to identify conceptually related tables, considering the similarity between both context sections as well as a table-to-context similarity between one table and the context associated with the other table (Yakout et al., 2012). Pimplikar et al. further extend this simple measure to enable collective matching that incorporates the table and context into a single similarity score. For a set of query terms, the extended measure splits the terms into two sections, matching one section to the table content (*inSim*) and the other to the context (*outSim*). The objective is to find the best separation of terms that returns the highest overall match (Pimplikar et al., 2012).

In summary, the context of a table is frequently recognized as an important resource in table understanding and integration tasks proposed in the literature. However, it is often regarded only as a general indicator for the overall meaning of a table and rarely studied in more detail to provide attribute-specific information. Furthermore, analysis has shown that the verbosity of large context segments often introduces noise that leads to inaccurate assumptions about the table content (Pimplikar et al., 2012). Yet, the relevance of a context section with respect to a specific table has received only limited attention so far. In this chapter, we address these shortcomings. In the subsequent sections, we study the relevance of contextual information as well as the potential for extracting attribute-specific information from the context.

5.3 CONTEXT RELEVANCE

As highlighted previously, context segments often provide useful information about the content of a table. However, a Web page or a public dataset can cover many different aspects of the main topic, while an individual table is often only related to a specific aspect. Consequently, only a subset of the document content or dataset description is actually related to the table. For the task of table under-



(a) Occurrence (b) Scores

Figure 5.1: Results of user survey, evaluating the context of Wikipedia tables.

standing, including other context that is not directly related to the table means introducing noise, which can significantly affect decisions based on the context information, such as in table matching or table search. Therefore, our goal in this section is to evaluate the relevance of context segments with respect to a specific table in order to reduce noise.

In Section 5.1, we already noted that some context segments are table-specific, including the title and caption of a table or resource metadata, while other segments are more generally related to the overall document or dataset, especially headlines and large text segments. The assumption is that these general segments are most likely to introduce noise (Pimplikar et al., 2012).

To gain a better understanding of the availability and quality of context information on the Web, and to see if we can confirm this intuition about large text segments and headlines, we conducted an initial user survey. First, we extracted different context segments for a sample of $\sim 170,000$ tables from English Wikipedia pages. Figure 5.1(a) presents the relative occurrence of different types of context on these Web pages. As context, we considered captions, text segments, page and section headlines, and links to other pages (i.e. only the anchor text). For this survey, we limited the text segments to text surrounding the table (in the same section of the Wikipedia article) and links to those also appearing in the same section as the table. In general, most Wikipedia articles contained some text somewhere on the page, while only about one third of all pages contained text in the same section as a table. Overall, the statistics show that table-specific context is significantly less frequent than general context information.

For a selection of 100 tables (with different context types equally distributed), we asked users to evaluate their comprehension of the table content with and without context information. For only 9% of tables, the users were able to understand the table content without context information, while additional information enabled the users to comprehend up to 90% of the tables fully or partially. To evaluate the individual context segments, we further asked users to score each segment based on how useful the provided information was for helping them understand the table content. Possible scores ranged from 1 (not useful) to 9 (very useful). Figure 5.1(b) illustrates the results of the survey. While the scores varied greatly for all segments, the results indicate that table-specific information, especially captions, provide a better description of the table content. Text segments, although limited to the same section of the Wikipedia article, were less helpful for the users, as were headlines.

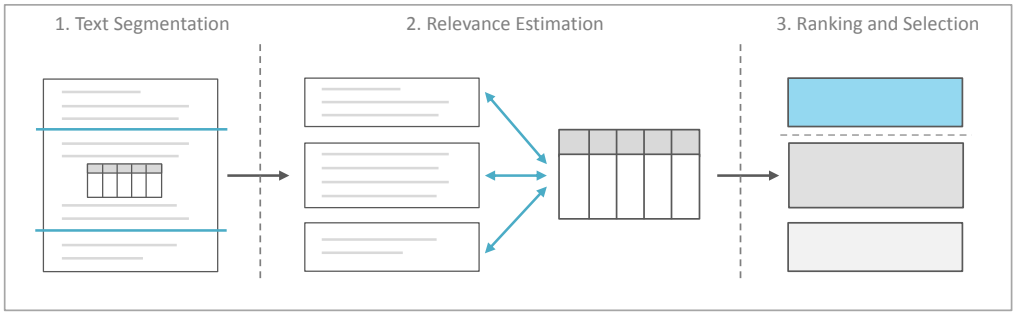


Figure 5.2: Overview of the paragraph selection task.

This initial study reveals that while additional context information is useful and important in order to understand the content of a table, table-specific context that provides relevant information is rare on the Web. Instead, it is much more common to find general headers and large text segments, which also contain a lot of irrelevant or misleading information.
As large text segments are the main source for noise in the context of a table, we focus on these segments. In order to improve the overall quality of context as an indicator of the meaning and intention of a table, we need to significantly reduce the noise. Consequently, a more detailed analysis of these text segments is required to distinguish between relevant and irrelevant information. To address this challenge, we propose an approach based on text segmentation and similarity estimation to evaluate the relevance of context information.

5.3.1 Problem Statement

We view the challenge of reducing the noise in the table context as a paragraph selection problem. Consequently, the objective is to identify paragraphs in long text segments which are semantically related or relevant to a table. We can split this problem into three subtasks, as illustrated in Figure 5.2.

1. First, the text is decomposed into topically coherent paragraphs. Selecting the best segmentation granularity is important, as the paragraph size can affect the selection process. If paragraphs are too large, we run the risk of retaining noise in the context. Yet, if paragraphs are too small, they do not provide enough content to make an informed decision regarding their relevance to the table.

2. Second, a similarity measure is used to match each paragraph to the table in question in order to evaluate its relevance. Often, the overlap between table content and context information is very limited. As a result, it is important to select an appropriate measure to estimate the relevance of the paragraphs.

3. And finally, all paragraphs are ranked according to their relevance and an appropriate threshold is determined to filter out irrelevant, noisy paragraphs. When selecting a threshold, we face a trade-off. If the threshold is too high, we potentially miss relevant context information, such as constraints or hidden attributes, affecting the recall of search or integration tasks. Yet, if we

set the threshold too low, we retain noise in the context, potentially affecting the precision of table-based applications.

5.3.2 Text Segmentation

The objective of the text segmentation step is to split long text sections into semantically coherent segments or paragraphs. Text segmentation is an important procedure within many text processing applications, such as passage retrieval or text summarization. As a result, various text segmentation algorithms have been proposed in the literature, including algorithms addressing lexical cohesion (Hearst, 1997), topic detection and tracking algorithms (Allan, 2002), as well as probabilistic models (Beeferman et al., 1999). Furthermore, we can distinguish between linear and hierarchical segmentation approaches.
In this thesis, we employ a linear text segmentation approach similar to *TextTiling* (Hearst, 1997), which detects shifts in topics by measuring the change in vocabulary within the text. Using a sliding window approach, vocabulary changes are detected by measuring the coherence between adjacent text sections. Significant changes in coherence determine the position of break points, as they indicate topic shifts. In detail, the algorithm proceeds as follows:

1. **Coherence Scores:** To measure the coherence, the text is first tokenized and split into smaller units. Common units are sentences or pseudo-sentences, i.e. token sequences of fixed length. While sentences provide for more natural boundaries, pseudo-sentences ensure that sections of equal size are compared. Two adjacent blocks of size b (in text units) are used to measure the change of vocabulary, as illustrated in Figure 5.3. A text similarity measure, such as the cosine similarity of the term frequency vectors, determines the coherence score s_c at the gap between both blocks. Sliding through the text with a step size of one text unit (sentence or pseudo-sentence), the coherence is measured throughout the text, resulting in a sequence of coherence scores. Low scores indicate potential topic shifts.

2. **Smoothing:** Before identifying the break points in the text, the sequence of coherence scores is smoothed to reduce the impact of small variations in coherence. In this thesis, an iterative moving average smoothing is applied.

3. **Depth Scores:** To identify suitable break points, the gaps of interest are the locations of the local minima of the coherence sequence. The significance of a topic shift is indicated by the depths of a local minimum compared to the coherence of neighboring text sections. This depth score s_d is defined in Equation 5.1 as the sum of the differences in coherence between local minimum i and the closest local maxima before (m_-) and after (m_+) the minimum, respectively.

$$s_d(i) = s_c(m_-) + s_c(m_+) - 2 \cdot s_c(i) \tag{5.1}$$

4. **Break Points:** As only significant vocabulary changes are likely to represent topic shifts in the text, a threshold is defined to filter out insignificant changes. Only gaps with a depth score $s_d \geq \mu - t \cdot \sigma$ are selected as break points. t is an adjustable threshold parameter, while μ and σ are the mean and standard deviation of the depth scores, respectively. In some cases, the resulting break point require further adjustment. If pseudo-sentences are used and a break point

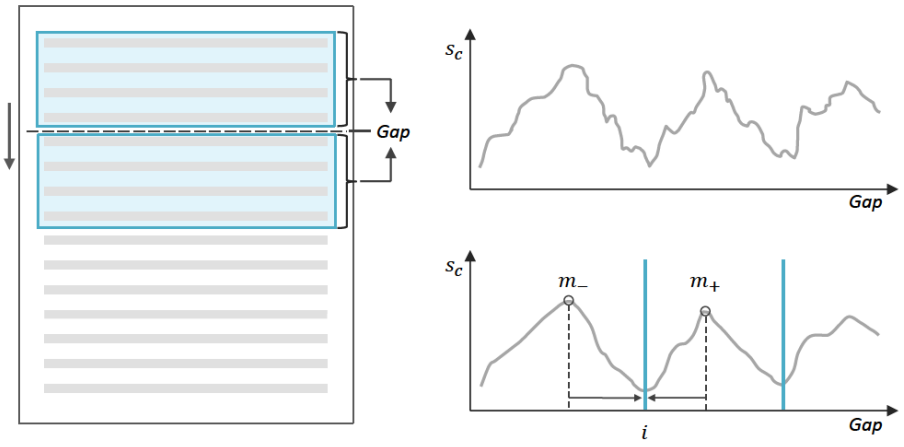


Figure 5.3: Linear text segmentation in *TextTiling* algorithm.

lies within a sentence, the next sentence break before or after the break point is used instead. Similarly, if the source text contained structural information, such as paragraphs, break points can be adjusted to fall on paragraph breaks nearby.

There are several parameters in the text segmentation algorithm that impact the quality of the returned segments, including the unit and block sizes, l and b, as well as the threshold parameter t. In addition, the selection of an appropriate similarity measure as well as smoothing technique also influence the number and location of break points. The optimal parameters depend on the characteristics of the text corpus.

To illustrate the impact of the main parameters, block size b and threshold parameter t, we evaluate the text segmentation algorithm on the standardized *Choi* dataset, a synthetic dataset frequently used as a benchmark for linear text segmentation (Choi, 2000). In the dataset, each text consist of several segments, with segment sizes varying between 3 and 11 sentences. For our evaluation, we consider 100 documents with diverse segment sizes. As the similarity measure, we use the cosine similarity of the term frequency vectors for each block. To evaluate the segmentation quality, we consider the *WindowDiff* evaluation metric proposed by Pevzner et al. (2002). The metric measures the probability that randomly selected (pseudo-) sentences that are k sentences apart, are *inconsistently* classified. Generally, k is estimated as half of the average text segment size in the reference segmentation. The *WindowDiff* value between a reference segmentation r and a test segmentation t is calculated as defined in Equation 5.2. For every pair of sentences that are k sentences apart, we consider the number of segment boundaries that lie between the two sentences, denoted as $b()$, as illustrated in Figure 5.4. If the number in the reference segmentation differs from the number in the test segmentation, the score is increased by one. Finally, the score is divided by the number of measurements taken.

We evaluate the text segmentation algorithm for various parameter settings and average the WindowDiff scores across all 100 documents. The results of the segmentation based on sentences as the basic text units are shown in Figure 5.5(a), while the result achieved with pseudo-sentences of length $l = 8$ words are shown in Figure 5.5(b). The WindowDiff metric returns a value in the range $[0, 1]$, with a lower score indicating a better segmentation. The results of our experiments show that the

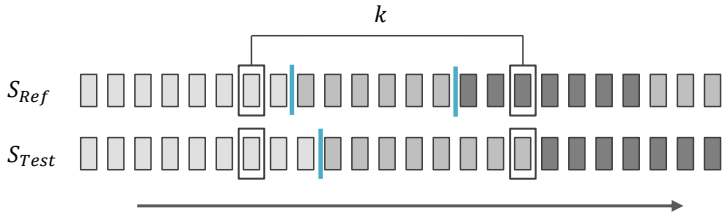
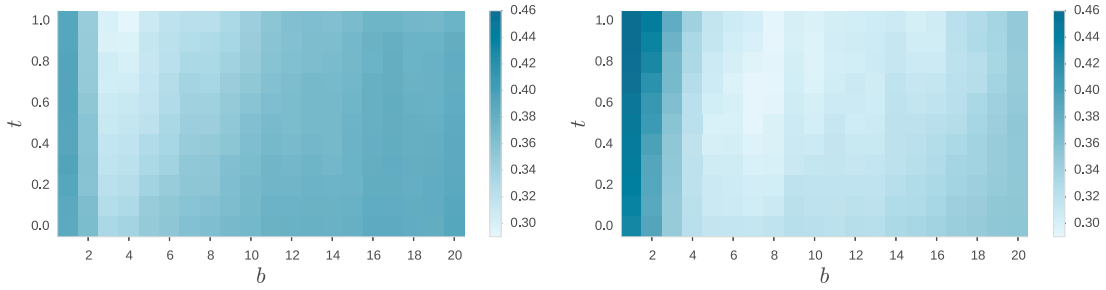


Figure 5.4: Evaluation of text segmentation in WindowDiff metric.



(a) Sentences

(b) Pseudo-Sentences

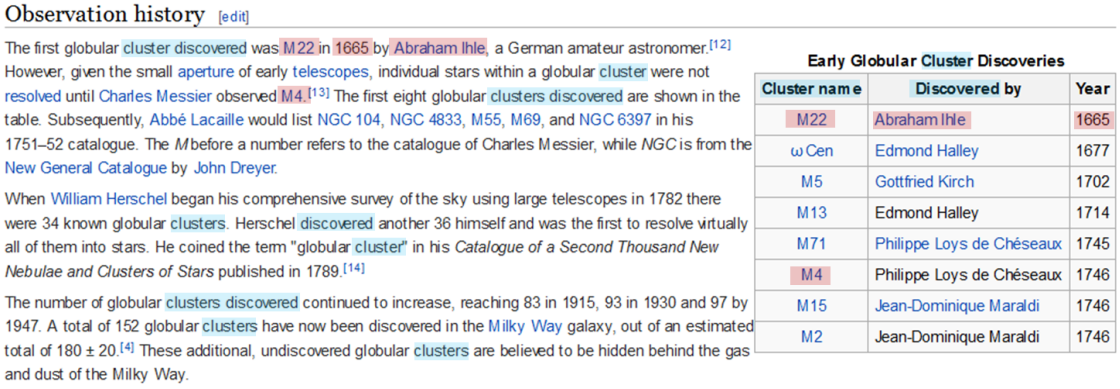
Figure 5.5: WindowDiff scores with respect to block size b and threshold parameter t.

segmentation algorithm is very sensitive to the values of the different parameters. Using sentences as the smallest units, we achieve the best score for the settings $b = 4$ and $t = 1.0$. In contrast, for pseudo-sentences we get similar results for various parameter settings, with the best scores achieved for $6 \leq b \leq 8$ and $0.4 \leq t \leq 0.8$. Selecting the optimal parameters for a segmentation task is challenging, as the parameters must be adjusted to the characteristics of the dataset.

$$WindowDiff(r,t) = \frac{1}{N-k} \sum_{i=1}^{N-k} (|b(r_i, r_{i+k}) - b(t_i, t_{i+k})| > 0) \tag{5.2}$$

5.3.3 Word-based Similarity

After splitting the context into smaller topical sections, our next goal is to evaluate the relevance of each segment with respect to the table content. Treating both the table and the context as a bag of words, i.e. assuming independence between words, we first focus on word-based similarity measures to estimate the relevance. The assumption is that if words from the table content, such as attribute labels or cell entries, are frequently mentioned in the context, it is very likely that the text describes the table (see Figure 5.6). Processing tables as a loose collection of words is a common approach, for example to find related tables (Cafarella et al., 2009) or to retrieve tables that match a user query (Pimplikar et al., 2012). Incorporating the table structure, which often provides implicit information about the dependencies between table entries, is difficult, because table structures are very heterogeneous and there are no general rules that apply to all tables. For the same reason, we also focus on word-based matching between tables and context sections.



Observation history [edit]

The first globular cluster discovered was M22 in 1665 by Abraham Ihle, a German amateur astronomer.[12] However, given the small aperture of early telescopes, individual stars within a globular cluster were not resolved until Charles Messier observed M4.[13] The first eight globular clusters discovered are shown in the table. Subsequently, Abbé Lacaille would list NGC 104, NGC 4833, M55, M69, and NGC 6397 in his 1751–52 catalogue. The *M* before a number refers to the catalogue of Charles Messier, while *NGC* is from the New General Catalogue by John Dreyer.

When William Herschel began his comprehensive survey of the sky using large telescopes in 1782 there were 34 known globular clusters. Herschel discovered another 36 himself and was the first to resolve virtually all of them into stars. He coined the term "globular cluster" in his *Catalogue of a Second Thousand New Nebulae and Clusters of Stars* published in 1789.[14]

The number of globular clusters discovered continued to increase, reaching 83 in 1915, 93 in 1930 and 97 by 1947. A total of 152 globular clusters have now been discovered in the Milky Way galaxy, out of an estimated total of 180 ± 20.[4] These additional, undiscovered globular clusters are believed to be hidden behind the gas and dust of the Milky Way.

Early Globular Cluster Discoveries

Cluster name	Discovered by	Year
M22	Abraham Ihle	1665
ω Cen	Edmond Halley	1677
M5	Gottfried Kirch	1702
M13	Edmond Halley	1714
M71	Philippe Loys de Chéseaux	1745
M4	Philippe Loys de Chéseaux	1746
M15	Jean-Dominique Maraldi	1746
M2	Jean-Dominique Maraldi	1746

Figure 5.6: Words matching between table and context [1]. Words from the table header are marked in blue, while words from the table body are marked in red.

Word-based similarity measures generally consider the frequency of terms as well as their significance to evaluate the similarity between text segments. Regarding a table as a loose collection of words, we face several issues. First of all, tables present information in a compact format, with most textual entries limited to words or short phrases and some of the information represented implicitly through the semantics of the table structure. Compared to text, tables contain significantly less explicit information, leading to very sparse term vectors. Second, the frequency of terms in a table is not representative of their importance regarding the tables main topic. Attribute labels, which are designed to describe the table content, generally only appear once, while some attribute values can appear numerous times due to redundancy in the table.
These characteristics are very similar to the characteristics of keyword queries in text retrieval systems. Compared to a longer text document, the term vector of a query is also very sparse, with little or no repetition of terms. Consequently, we can regard a table as a long keyword query. In the literature, a wide range of retrieval functions have been proposed, which score documents based in their relevance to the query, taking into account the sparsity of the query. These retrieval functions present one option to identify relevant context segments for Web tables.
Considering that we also have a significant number of large tables on the Web, which feature a term count similar to that of the average context paragraph, we can consider text similarity measures as another option to evaluate context relevance. These symmetric measures regard both, the table and the context segment, as a bag of words and calculate a similarity score. In the Web table literature, text similarity measures, such as the cosine similarity of weighted term vectors, have been applied to match tables to other tables or text segments (Cafarella et al., 2009).
Both approaches, using retrieval functions or symmetric text similarity measures, present viable options for table-to-context matching. Considering the diverse characteristics of Web tables, it is difficult to favor one approach over the other. To gain a better understanding of the functionality and behavior of these measures with respect to Web tables, we conduct a comparative study, analyzing different measures proposed in the literature for a test set of Web tables. For the evaluation, we use a set of 30 tables extracted from the English Wikipedia along with their respective pages. To retrieve context

[1] Source: `https://en.wikipedia.org/wiki/Globular_cluster`

paragraphs, we utilize the original structure of the Wikipedia articles, considering headlines as natural topic boundaries. We manually judged each resulting paragraph as either *relevant* or *not relevant*. After applying the different measures to score the context paragraphs, we use the following metrics to evaluate the suitability of the similarity measures:

- **Mean Reciprocal Rank (MRR):** The mean reciprocal rank, as defined in Equation 5.3, evaluates the quality of the paragraph rankings produced for a set of tables T. The quality of a single ranking is measured using the *reciprocal rank*, defined as the multiplicative inverse of the rank of the first relevant paragraph in the ranked list.

$$MRR = \frac{1}{|T|} \sum_{i=1}^{|T|} \frac{1}{rank_i} \tag{5.3}$$

- **Mean Average Precision (MAP):** Since we can also have multiple relevant paragraphs for a table, we also consider the mean average precision, which takes the rank of all relevant paragraphs into account. The average precision (AveP) for a table is calculated by considering, for each relevant paragraph, the precision at the rank of the paragraph and computing the average of these precision values. This measure takes into account the order in which relevant and non-relevant paragraphs are ranked, returning the highest value if all relevant paragraphs are ranked before any non-relevant paragraph.

$$MAP = \frac{1}{|T|} \sum_{t \in T} AveP(t) \tag{5.4}$$

Retrieval Functions

First, we consider several state-of-the-art retrieval functions to estimate the relevance of context segments with respect to a table. The objective of a retrieval function is to rank documents based on their relevance to a query, which generally is a list of keywords or phrases. Different retrieval functions use different techniques to address the importance of query terms and the length of the document. As retrieval functions we consider the following established techniques:

1. **TF-IDF weighting scheme:** As the first retrieval function, we use a simple weighting scheme that estimates the importance of query (or table) terms with a weighting function w, then scores paragraphs using the sum of the weights of query terms appearing in the paragraph, as specified in Equation 5.5. A popular weighting function in text retrieval applications is the *TF-IDF* score, which incorporates the frequency of a term in the document in question (TF) as well as the inverse document frequency (IDF), i.e. the total number of documents the term appears in (in the collection). Different variants to compute the TF and IDF components have been proposed in the literature (Zobel et al., 1998). Here, we use the variant specified in Equation 5.6. Note that the term weight $w_D(t) = 0$, if term t does not appear in document D. The term frequency $TF_D(t)$ represents how prominent a term is in a single document, while the inverse document frequency is defined as a global score over the entire document corpus of size N. With context paragraphs acting as documents in our case, we can derive different ways to define the IDF score. First, we can define the document frequency over paragraphs and set N to the total number of paragraphs in the table collection (denoted as IDF(p)). Alternatively, we can score

terms considering complete table contexts as documents, setting N to the number of context instances in the collection (denoted as IDF(d)). Finally, we can define IDF as a local score instead of a global score, taking only the passages in the context of a table into account. As we only match the table terms to the paragraphs of the table's context, not to paragraphs in the context of other tables, we define the IDF score per table context, as the number of paragraphs in the context divided by the number of paragraphs that contain term t. We refer to this score as the *Inverse Paragraph Frequency (IPF)*.

$$S_{TF-IDF}(T, D) = \sum_{t \in T} w_D(t) \tag{5.5}$$

$$w_D(t) = (1 + \log TF_D(t)) \cdot \log \left(\frac{N}{DF(t)} \right) \tag{5.6}$$

2. **Language Models:** As a second retrieval approach, we consider language models, probabilistic models that reflect the distribution of terms in documents (Ponte et al., 1998). We consider unigram models, which assume independence between terms. Retrieval based on language models ranks documents based on the likelihood of the query (or table T) given the document model M_D, as defined in Equation 5.7 .

$$S_{LM}(T, D) = P(T|M_D) \approx \prod_{t \in T} P(t|M_D)^{TF_T(t)} \tag{5.7}$$

Often, the logarithm of $P(T|D)$ is computed, which transforms the product of term probabilities into a sum. To avoid issues for query terms that do not appear in a document, so-called *smoothing* techniques have been proposed in the literature (Zhai et al., 2004). These smoothing techniques utilize a *collection model C* to account for *unseen* terms and a smoothing parameter to control the influence of the document and collection models. Here, we consider the two most common smoothing techniques, Jelinek-Mercer smoothing, denoted as P_λ in Equation 5.8, and Dirichlet smoothing, denoted as P_μ in Equation 5.9.

$$P_\lambda(t|D) = (1 - \lambda) \cdot P(t|D) + \lambda \cdot P(t|C) \tag{5.8}$$

$$P_\mu(t|D) = \frac{TF_D(t) + \mu \cdot P(t|C)}{(\sum_{t' \in D} TF_D(t')) + \mu} \tag{5.9}$$

3. **Okapi BM25:** As the last group of retrieval functions, we consider *Okapi BM25*, a probabilistic retrieval function frequently used for text retrieval (Robertson et al., 1996). To score a document, the scoring function, specified in Equation 5.10, takes into account the frequency of a term both in the query and the document as well as the inverse document frequency of the term. In addition, the size of the document $|D|$ as well as the average document size of the collection $avgdl$ are included to correct the score depending on the size of the document at hand. Parameters k_1, k_3 and b can be set to adjust the scoring function to the characteristics of the documents and queries. The complete function in Equation 5.10 is predominantly used for long queries, while the first factor of the product is often ignored for short queries. Here,

Table 5.1: Evaluation of retrieval functions to estimate the relevance of context segments.

Retrieval Function	MRR	MAP
Vector Space Model		
· TF	0.944	0.816
· TF-IPF	0.961	0.829
· TF-IDF (Documents)	0.951	0.827
· TF-IDF (Passages)	0.950	0.839
Language Models		
· Dirichlet Smoothing	**0.978**	**0.865**
· Jelinek-Mercer Smoothing	0.961	0.844

Retrieval Function	MRR	MAP
Okapi BM25 (short)		
· No IDF	0.977	0.829
· IPF	0.977	0.827
· IDF (Documents)	0.934	0.827
· IDF (Passages)	0.958	0.827
Okapi BM25 (long)		
· No IDF	0.961	0.837
· IPF	0.961	0.835
· IDF (Documents)	0.951	0.844
· IDF (Passages)	0.942	0.831

we consider both variants, as well as the different IDF specifications that we also used in the TF-IDF weighting scheme.

$$S_{BM25}(T, D) = \sum_t \frac{TF_T(t) \cdot (k_3 + 1)}{TF_T(t) + k_3} \cdot \frac{TF_D(t) \cdot (k_1 + 1)}{TF_D(t) + k_1 \cdot bl_D} \cdot IDF(t) \tag{5.10}$$

$$bl_D = \left(1 - b + b \cdot \frac{|D|}{avgdl}\right) \tag{5.11}$$

To determine the suitability of these retrieval functions, we evaluate each retrieval function (with different parameter settings, if applicable) on the test set. To analyze the sensitivity of each approach to table size, we split the test set into small, medium sized and large tables, with table sizes of less than 20 terms, between 20 and 200 terms and more than 200 terms, respectively. In Table 5.1, we present the best overall MRR and MAP scores for each approach. The detailed evaluation for different table sizes is included in Appendix D.1.
Overall, all retrieval functions achieve high scores with only little variance across the different functions. For our test set, the highest MMR and MAP scores are achieved using retrieval functions based on language models with Dirichlet smoothing. The results indicate that more sophisticated retrieval functions such as language models or BM25 are better suited for the identification of relevant context than the simple weighting scheme. Between the different table sizes, we do not observe major differences.

Similarity Metrics
In addition to various document retrieval functions, we compare various symmetric text similarity measures. Probably the most common text similarity measure is the cosine similarity of weighted term vectors, which has also been used frequently in the Web table recovery literature. Besides this measure, we also consider two alternative measures, proposed by Whissell et al. (2013), which rep-

resent symmetric variants of popular retrieval functions. In detail, we consider the following text similarity measures in our study:

1. **Cosine TF-IDF:** The cosine similarity represents a similarity measure frequently used in connection with the vector space model, which models texts or documents as vectors of term weights, one for each term in a dictionary (Salton et al., 1988). The cosine similarity, as specified in Equation 5.12, measures the cosine of the angle between the vectors, with a small angle indicating similar documents. The most common weighting functions for the terms in the documents include the TF-IDF score and its variants.

$$S_{TF-IDF}(T, D) = \frac{\vec{w}_T \cdot \vec{w}_D}{\|\vec{w}_T\| \cdot \|\vec{w}_D\|} \tag{5.12}$$

2. **Symmetric Language Models:** As the first alternative to the cosine similarity, we consider a symmetric similarity measure based on language models proposed by Whissell et al. (2013). For a table T and a context paragraph D, the similarity measure is defined as specified in Equation 5.13. The probability estimates equal the estimates used for retrieval. Consequently, the same smoothing techniques can be applied, such as Jelinek-Mercer or Dirichlet smoothing.
Using language models, it is often very likely for two long documents to feature many similarities simply due to terms that are very frequent in the language and, thus, appear in most documents. To account for this scenario, Whissell et al. incorporate $\log P(*|C)$ to model chance, where C is the collection or background model that reflects the general frequency of terms in the language.

$$S_{LM}(T, D) = \log(P(D|T)) - \log(P(D|C)) + \log(P(T|D)) - \log(P(T|C)) \tag{5.13}$$

3. **Symmetric Okapi BM25:** As the second alternative, Whissell et al. also propose a symmetric variant of the Okapi BM25 retrieval function, as specified in Equation 5.14. To ensure symmetry, the first factor of the retrieval function in Equation 5.10 is replaced by a factor that equals the second factor, utilizing the same parameters k_1 and b. Again, we can use different variants of IDF or omit the score.

$$S_{BM25}(T, D) = \sum_t \frac{TF_T(t) \cdot (k_1 + 1)}{TF_T(t) + k_1 \cdot bl_K} \cdot \frac{TF_D(t) \cdot (k_1 + 1)}{TF_D(t) + k_1 \cdot bl_D} \cdot IDF(t) \tag{5.14}$$

Similar to the retrieval functions, we evaluated each similarity measure on the test set to determine their suitability. The overall best MRR and MAP scores are presented in Table 5.2, while the detailed analysis for different table sizes is included in Appendix D.2. In general, the symmetric similarity measures perform equally well as the retrieval functions, with only little variance between the different measures. Again, the measures based on language models are among the best candidates.
However, comparing the cosine and BM25 similarity measures, we can see that for our test set the cosine similarity with simple TF weights outperforms the more complex weighting scheme of BM25. For both approaches, we can also observe that inverse paragraph frequency (IPF) performs better than the other IDF variants.
The analysis of the various symmetric similarity measures for different table sizes shows slightly more variation. As expected, all measures achieve the highest scores for larger tables with more than 200 terms. Overall, symmetric text similarity measures produced results of high quality.

Table 5.2: Evaluation of text similarity measures to estimate the relevance of context segments.

Similarity Measure	MRR	MAP
Cosine Similarity		
· TF	0.977	**0.871**
· TF-IPF	0.961	0.843
· TF-IDF (Documents)	0.944	0.850
· TF-IDF (Passages)	0.936	0.854
Language Models		
· Dirichlet Smoothing	**0.978**	0.867
· Jelinek-Mercer Smoothing	0.961	0.859

Similarity Measure	MRR	MAP
Okapi BM25		
· No IDF	0.928	0.816
· IPF	0.950	0.819
· IDF (Documents)	0.930	0.840
· IDF (Passages)	0.928	0.813

5.3.4 Topic-based Similarity

If the vocabulary in the documents is large, word-based similarity measures operate in a very high-dimensional space, as the size of the vocabulary determines the dimensionality. In practice, computing the similarity in such a high-dimensional space can be very computationally expensive and impractical. Furthermore, the analyzed data becomes very sparse in such a high-dimensional space, which can impact the ability to identify similar documents. To address this dimensionality issue associated with word-based similarity measures, we consider *topic modeling* as an alternative. Instead of comparing tables and context segments at word level, where the size of the vocabulary determines the dimensionality, we compare the *topics* associated with the words, instead. Generally, the number of topics expected in a collection of documents is significantly lower than the size of the vocabulary, which reduces the dimensionality of the task. Consequently, each table and each text segment are represented by a k-dimensional topic vector, where k is the number of topics considered.
In addition to reducing the dimensionality, topic modeling also enables the identification of implicit matches between a table and the associated context, where both describe the same topic, but do not explicitly use the same terms to describe it. Such relations between tables and context cannot be detected via word-based matching.

Topic Modeling
Topic modeling aims to identify abstract topics in a document collection and to model each document based on its association with one or more of these topics. In general, the presence of a topic is determined from the frequency and co-occurrence of certain terms in the document, and, in most cases, a topic is represented as a probability distribution over all terms in the vocabulary.
In the literature, various algorithms have been proposed to model topics, including latent semantic indexing (LSI) (Deerwester et al., 1990), probabilistic latent semantic indexing (PLSI) (Hofmann, 1999) and latent dirichlet allocation (LDA) (Blei et al., 2003). Topic models have been applied successfully to a wide range of machine learning and data mining scenarios, including document retrieval (Wei et al., 2006) and text segmentation (Riedl et al., 2012). In this thesis, we focus on latent dirichlet allocation to model topics in tables as well as their context.

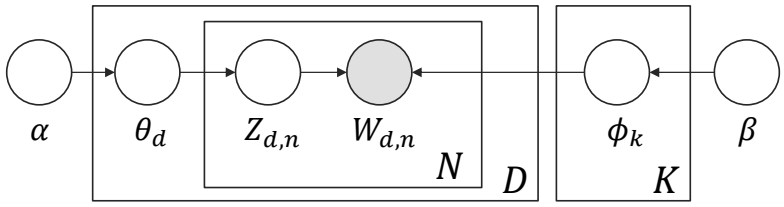


Figure 5.7: Plate notation of graphical model for smoothed LDA model. Nodes in the graph represent random variables, with shaded nodes representing *observed* variables and unshaded nodes representing *latent* variables. The rectangles denote replication in the graph.

Latent Dirichlet Allocation

Latent dirichlet allocation (LDA) is a generative probabilistic model initially developed by Blei et al. to model topics in text collections (Blei et al., 2003). LDA is based on a generative process that allows for documents to cover multiple topics. Figure 5.7 illustrates LDA as a graphical model in plate notation. The model incorporates topics K, documents D and words from a fixed vocabulary N. $W_{d,n}$ denotes the n-th word in document d. The topical structure in a corpus is modeled by random variables ϕ_k, θ_d and $Z_{d,n}$. Each topic k is described by a multinomial probability distribution ϕ_k over the term vocabulary. Furthermore, each document d is associated with a multinomial distribution θ_d which describes the topic proportions for the document. As each document can describe multiple topics, this distribution reflects the mixture of topics in the document. Finally, $Z_{d,n}$ denotes the topic assignment for word n in document d. Variables α and β are hyper-parameters. Based on this graphical model, the topical structure of a corpus can be derived by computing the conditional probability of the topic structure given the observed words in the documents (see Equation 5.15). However, computing this posterior distribution is intractable for large document collections, thus requiring approximate solutions, such as Gibbs sampling or mean-field variational inference (Blei et al., 2009).

$$p\left(\phi_{1:K}, \theta_{1:D}, z_{1:D} | w_{1:D}\right) = \frac{p\left(\phi_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}\right)}{p\left(w_{1:D}\right)} \tag{5.15}$$

LDA models documents as a mixture of topics instead of describing a single topic. In its simplest form, it assumes independence between words in the document as well as independence between topics in a document. Extended models have been proposed, which relax these assumptions, such as the *topical N-grams* model (Xuerui Wang et al., 2007). Furthermore, the simple model requires the number of topics to be modeled (K) as input parameter. The best parameter value depends strongly on the corpus at hand.

Similarity Measures

For each document or document section, the LDA inference generates a vector of *topic proportions* $\widehat{\theta}_d$, which represents a discrete probability distribution over all topics. Consequently, in order to measure the similarity between the topical representations of two documents, we can apply measures that quantify the similarity between probability distributions.

Blei et al. (2009) suggest the *Hellinger distance*, defined in Equation 5.16. As $dist_H$ returns a value in the range $[0, 1]$, we can derive a similarity measure, using $1 - dist_H$.

$$dist_H(d_1, d_2) = \frac{1}{\sqrt{2}} \sqrt{\sum_{k=1}^{K} \left(\sqrt{\widehat{\theta}_{d_1,k}} - \sqrt{\widehat{\theta}_{d_2,k}} \right)^2} \tag{5.16}$$

Another common measure is the *Kullback-Leibler (KL) divergence*, which, in its original form, is not strictly a metric, as it is not symmetric. The KL divergence of probability distribution P_2 from P_1 is denoted as $D_{KL}(P_1||P_2)$. With respect to topic proportions, it is defined as follows:

$$D_{KL}(d_1||d_2) = \sum_{k=1}^{K} \widehat{\theta}_{d_1,k} \ln \frac{\widehat{\theta}_{d_1,k}}{\widehat{\theta}_{d_2,k}} \tag{5.17}$$

Various symmetric variants of the KL divergence have been proposed to derive a distance metric. The sum of $D_{KL}(d_1||d_2)$ and $D_{KL}(d_2||d_1)$ is frequently used. From this symmetric distance measure, we can derive a similarity score as defined in Equation 5.18.

$$sim_{KL}(d_1, d_2) = 10^{-(D_{KL}(d_1||d_2) + D_{KL}(d_2||d_1))} \tag{5.18}$$

Evaluation

To evaluate the suitability of LDA to estimate the relevance of table context, we use the test set of 30 tables from the previous section. We train the LDA model using a corpus of $1,000$ English Wikipedia articles (from which the test tables were selected). For the hyper-parameters, we use the settings often recommended in the literature (see, for instance, the work of Wei et al. (2006)), with $\beta = 0.01$ and $\alpha = \frac{50}{K}$, where K is the number of topics considered in the model. We varied the number of topics in our experiments, but limit the results presented here to $K \in \{200, 500, 1000\}$.

For each table and each context segment, we infer a topic distribution using the trained LDA model and measure the similarity between the topic distribution of the table and the topic distribution of each segment in the context of the table. Similar to the previous evaluation, we use the mean reciprocal rank (MRR) and the mean average precision (MAP) to evaluate the rankings. Again, we also study tables of different sizes.

Table 5.3 shows the overall MRR and MAP scores, while the detailed evaluation is included in Appendix D.3. The scores achieved on our test set with different topic counts indicate that topic modeling is significantly less effective in estimating the context relevance, compared to word-based matching techniques. We observe only very little variation for different topic counts.

In our analysis, we can identify two possible reasons for the significantly lower results. The first issue is the table size. It appears that most tables in our test set are too small, i.e. they contain too few terms, in order to enable a meaningful inference of topic distributions. A small document size is a general issue of LDA inference, previously highlighted in a comprehensive study by Tang et al. (2014). An analysis of tables of different sizes confirms this assumption, as both MRR and MAP scores improve with increasing table size.

The second issue is the topic count and granularity. In an open domain scenario, such as the Web, we face a huge number of possible topics, which is replicated, although on a slightly smaller scale, on Wikipedia. Using very general topics reduces the number, however, in order to distinguish between the topics of paragraphs of the same document, we require very detailed topics. Consequently, it is

Table 5.3: Evaluation of latent dirichlet allocation to estimate the relevance of context segments.

Settings	MRR	MAP
$\mathbf{sim_H}$		
· $K = 200$	0.549	0.462
· $K = 500$	0.569	0.437
· $K = 1000$	0.538	0.457

Settings	MRR	MAP
$\mathbf{sim_{KL}}$		
· $K = 200$	0.569	0.485
· $K = 500$	0.549	0.433
· $K = 1000$	0.555	0.449

very difficult to model the subtle differences between paragraphs, if the overall corpus is already very heterogeneous and diverse. A large number of topics is another challenge for LDA inference highlighted by Tang et al. (2014). One possible approach to address this issue, is to cluster the documents into more coherent subsets and train a separate LDA model for each cluster.
For our test set, directly applying LDA to Web tables and their respective context segments does not offer any benefits over the word-based similarity measures. Therefore, we do not further consider this approach in the reminder of this chapter.

5.3.5 Filter and Threshold - Context Selection

After evaluating the relevance of each context section with respect to the table, using a retrieval function or symmetric text similarity measure, we can retrieve a ranked list of context sections. In the final step, we need to decide which context sections to keep for subsequent processing and which sections to discard as irrelevant or noisy. Consequently, we require a threshold for the relevance score.
Finding the optimal threshold for a large collection of tables and their respective contexts is very challenging, as the Web pages can have very different characteristics. In some cases, only a small section on the Web page is related to the content of a table, while in other cases the entire Web page can be regarded as relevant. Furthermore, the similarity measures are not always able to make a clear distinction between relevant and irrelevant context. Thus, when selecting a relevance threshold, we face a trade-off between eliminating noise in the context and missing potentially relevant information. To address this trade-off, we consider two alternative threshold specifications:

- **Rank-based Threshold:** A rank-based threshold is a popular selection approach in retrieval systems. Instead of considering the value of the relevance score, context segments are regarded as relevant based on their position in the ranked list of all context sections. Only the top k sections are retrieved.

- **Score-based Threshold:** In contrast, the score-based threshold is not associated with a fixed position in the ranked list, and, instead, takes the variance of the relevance scores across the context sections into account. In particular, the threshold is defined as follows: $\theta_{score} = \mu - t \cdot \sigma$, where μ and σ are the mean and standard deviation of the relevance scores, respectively.

While the rank-based threshold returns the same number of context sections for each table, the score-based threshold is different for each table. For each approach, we can adjust the threshold by varying

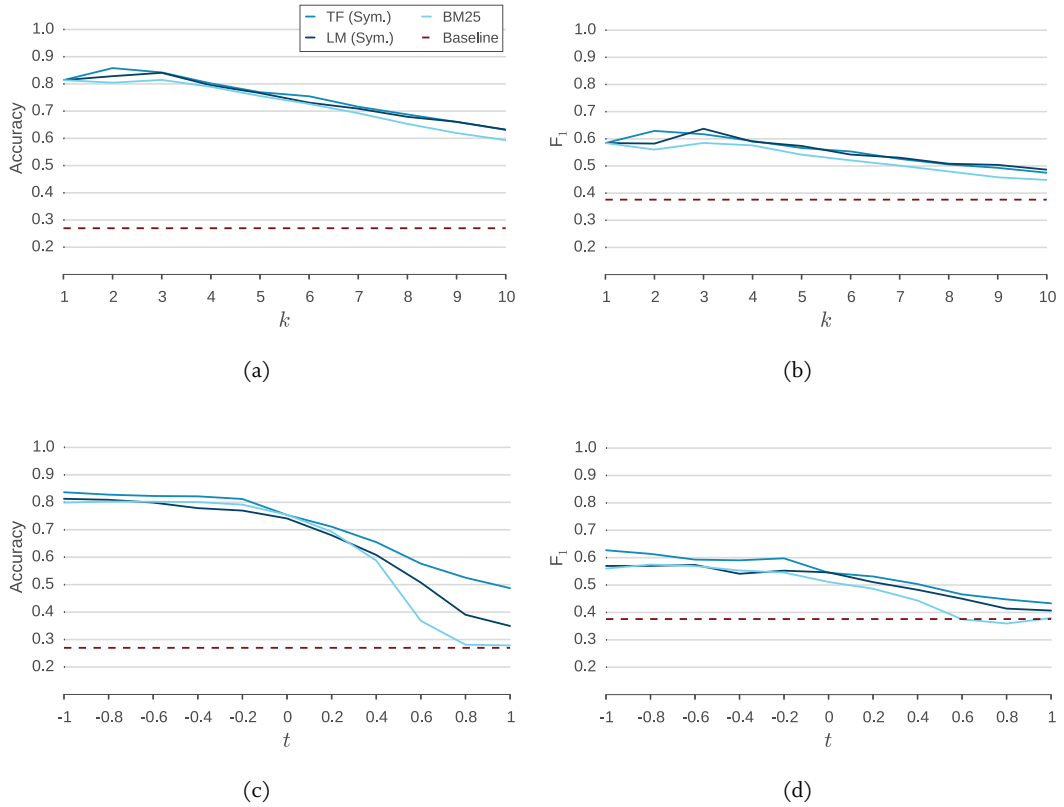


Figure 5.8: Average accuracy and F_1 measure of context selection using rank-based and score-based thresholds.

the parameters k and t, respectively. Using the test set of 30 Web tables and associated context sections, we can analyze the characteristic behavior of each threshold approach. Varying the threshold parameters, we measure the accuracy as well as the F_1 measure, averaged across all tables. *Accuracy* measures the percentage of correctly identified context sections, i.e. the number of relevant sections that have been retrieved as well as the number of irrelevant sections that have been discarded. The F_1 measure only considers the retrieved sections and takes into account precision and recall. The precision states how many of the retrieved sections are actually relevant to the table, while recall states how many of the relevant sections have been retrieved. Figure 5.8 shows the results. In our evaluation, we consider three different measures to compute the relevance scores of the context sections. As the different retrieval functions and similarity measures we studied in the previous section all produce very different relevance scores and rankings, the choice of scoring function can influence the quality of the retrieved context sections. For the experiments, we consider the cosine similarity of TF scores, a symmetric similarity score based on language models with Dirichlet smoothing (LM) as well as the BM25 retrieval function. As a baseline, we measure the accuracy and F_1 for the case where all context sections are retrieved. Consequently, a higher score indicates an improvement achieved through context selection.

Figures 5.8(a) and 5.8(b) show the results for the rank-based threshold approach for k in the range $[1, 10]$. There are only little differences between the various scoring functions. We can see an obvious improvement over the baseline, which is decreasing as more context is retrieved. The less significant improvement in F_1 measure indicates the weakness of a fixed rank-based threshold. With a fixed number of retrieved context sections, this approach does not adapt very well to the different characteristics of Web table context, where some tables have much less relevant context sections than others.
The results of the score-based threshold are presented in Figures 5.8(c) and 5.8(d), for parameter t in the range $[-1, 1]$. Here we can clearly see the impact of the scoring function, especially for higher values of t. Overall, the maximum accuracy and F_1 values that can be achieved with this threshold approach are very similar to those achieved with a rank-based threshold. The score-based threshold assumes some variation in the relevance scores of the context sections. However, if all context sections are equally relevant and receive very similar scores, the threshold discards some of the relevant sections, which is reflected in the F_1 scores.

5.3.6 Discussion

In this section, we have described a paragraph selection algorithm that aims to identify paragraphs in long text segments that are semantically related or relevant to a table. By focusing on relevant context, we hope to provide a more accurate description of the table content and reduce the noise that leads to incorrect table relevance decisions.
As part of this section, we have studied a variety to established retrieval functions and similarity measures to estimate the relevance of table context. While topic models are not very well suited for the task, word-based similarity measures have achieved very good results, given the heterogeneity and quality of real-world Web data. Various subsequent table processing tasks, such as the extraction of hidden attributes or constraints from the context, or the identification of semantically related tables, benefit from reducing the noise and topic drift in the context.
In the next section, we now utilize the table context to extract detailed information that directly addresses individual attributes in the table.

5.4 ATTRIBUTE-SPECIFIC CONTEXT ANNOTATION

For many Web table applications, such as table search or table integration, it is important to have an accurate understanding of the individual attributes described in a table. However, as we highlighted in Section 3.1, attribute labels in Web tables are often of low quality or too general to provide a suitable description of the attributes. Additionally, many tables provide no attribute labels at all.
To overcome this challenge, various techniques have been proposed in the literature to infer better attribute labels from external knowledge sources (based on the instances of an attribute) or by propagating labels from related tables with higher label quality. In Section 3.3.2 we give a detailed review of these techniques.
The context associated with a table on the Web has, so far, received only little attention with respect to understanding individual attributes. The context is frequently considered as an indicator for the general content of a table, but rarely as a source for attribute specific details (see also Table 3.3 in Section 3.3.2). However, the contextual description of the table content often contains phrases that

provide a sufficiently detailed specification of the attributes and can be used as alternative labels. We refer to these phrases extracted from the context and associated with an attribute in the table as *attribute-specific context annotations*.
In this section, we propose various techniques that enable the identification and extraction of relevant, attribute-specific information from the table context.

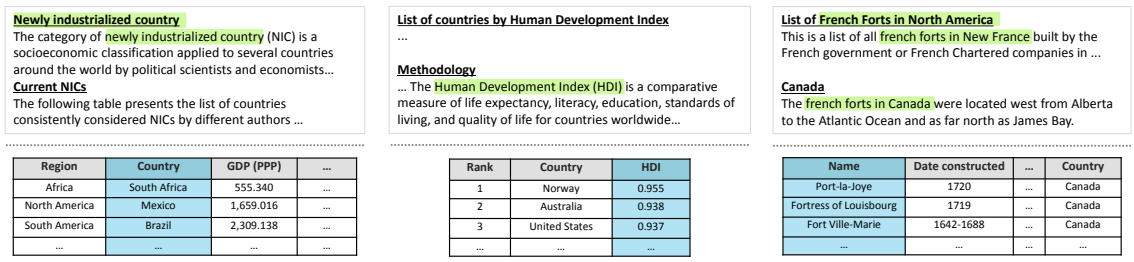
5.4.1 Problem Statement

In detail, we consider the following task. Given an attribute A in a table T, with a set of instances $\mathcal{I}_A$ and an original attribute label L_A (if available), our objective is to extract a set of alternative labels $\{L_{C_1}, \ldots, L_{C_n}\}$ from the context of T. Each label L_{C_i} should have the following properties: (1) $L_{C_i} \neq L_A$, and (2) L_{C_i} gives a valid description of attribute A. As the context of T, we consider all textual sources described in Section 5.1, including headlines, captions and surrounding text.
In most languages, there are many different ways to refer to the same thing (or attribute in our case), using linguistic variations of words or different syntactic patterns. Consequently, identifying all mentions of an attribute in the context is very challenging and generally requires a complex linguistic analysis of all words, their respective roles and subsequent meaning in the text.
To reduce the complexity of this task, we focus predominantly on noun phrases in the context. The majority of labels used to describe attributes in Web tables are either single nouns or noun phrases. As a result, considering only noun phrases as alternative labels, makes it easier to identify matches between the original label and the context and ensures that the extracted phrases can directly be used as suitable labels for the attribute in question.
In our approach, we consider two types of context annotation, *directly* and *indirectly related* phrases. An extracted label L_{C_i} is *directly related* to original label L_A, if $L_A \subset L_{C_i}$, meaning that the original label is contained in the extracted phrase. On the other hand, the label L_{C_i} is *indirectly related*, if we can derive an intermediate label L_I from attribute A so that $L_I \subset L_{C_i}$. For each annotation type, we consider several extraction and inference techniques, which are described in the subsequent sections.

5.4.2 Directly Related Context

As stated previously, a noun phrase in the context is considered *directly related* to an attribute in the table, if the terms used in the original attribute label appear in the phrase. We study two different variations of such directly related phrases: (1) qualifying noun phrases, i.e. noun phrases that extend the original noun phrase with additional details, and (2) noun phrases that represent expansions of acronyms or abbreviations found in attribute labels. In the subsequent sections, we give a detailed description of the extraction algorithms involved in identifying these direct matches.

Qualifying Noun Phrases
We are mostly interested in noun phrases in the context that provide a more detailed description of an attribute in the table, so-called *qualifying* noun phrases. Figure 5.9(a) shows an example. These noun phrases generally consist of a noun as the *head* of the phrase and a *dependent* that modifies the noun (Huddleston et al., 2002). Common dependents include attributive adjectives, as in "*modern* car", adjective phrases like "people *living in cities*", or noun adjuncts such as "*football* player". In the context, these dependents often contain supplementary or modifying information on the attribute in



(a) Qualifying Noun Phrase (b) Acronyms (c) Indirect Match

Figure 5.9: Example tables with corresponding context sections. Marked phrases (green) provide a detailed description of the selected attributes (blue) in each table.

question, that is not included in the original label, but is important for a correct interpretation of the attributes meaning.
In order to extract qualifying noun phrases from context sections, we apply an extraction algorithm with the following processing steps:

- **Candidate Extraction:** In a first step, we extract all noun phrase candidates from the context section. To do that, we apply a lexical parser that identifies the noun phrases and their respective head nouns. In addition, we parse all attribute labels, to identify those that contain a noun or noun phrase.
- **Label Matching:** For the direct matching approach, we search for mentions of the attribute labels in extracted noun phrase candidates. To account for different surface forms in the labels and phrases, such as singular and plural variants, we apply lemmatization in addition to case folding before matching. For the selection of relevant matches we follow a conservative matching strategy that requires the head of the attribute label to match the head of a candidate noun phrase. This approach eliminates matches where the attribute labels only appears in the dependent of the noun phrase. We found that a more generous strategy that allows such matches introduces too much noise.
- **Filter:** After selecting all potentially related noun phrases, we apply a final filter step to remove obvious false positives. A common issue we encountered is the mention of column instances in the context. If these instance names also match the attribute label, as is the case for the names *New York City* or *Mexico City* that match the attribute label *City*, for example, our algorithm extracts these instance names as qualifying noun phrases. We address this issue by simply discarding all candidate noun phrases that match instances in the table.

The extracted noun phrases are used as annotations to the attributes in question, to support search or integration of tables.

Acronyms and Abbreviations
To meet the spatial limitations of a Web page or simply out of convenience, attribute labels often contain acronyms or abbreviations, as illustrated in the example in Figure 5.9(b). Unless the abbreviated term is common enough to not require an explanation, which is often the case for units such as *km* or

Algorithm 5.1 Generate regular expression pattern for acronym A.

```
1:  function RegularExpression(A)
2:      rx = ""                                          ▷ Regular Expression rx
3:      for all c ∈ A do                                 ▷ Character c
4:          if c ≠ LastCharacter(A) then
5:              rx = rx + c+"[a-z]+[ ]+[A-Za-z ]*"
6:          else
7:              rx = rx + c+"[a-z]+[A-Za-z ]*"
8:          end if
9:      end for
10:     return rx
11: end function
```

mph, the extended form is usually mentioned somewhere in the context of the table. We refer to this longer form of an acronym or abbreviation as its *expansion*. In the example in Figure 5.9(b), *Human Development Index* represents the expansion of acronym *HDI*. These expansions are also useful phrases to provide a better description of the content of an attribute.
Expansions of acronyms and abbreviations can be detected using linguistic normalization techniques based on regular expressions. In particular, we incorporate the regular expressions proposed by Sorrentino et al. (2009).
For the extraction of appropriate acronym expansions, we propose the following algorithms. First, we identify candidate phrases in the context that match the following simple pattern, where *NP* stands for noun phrase:

$$NP(NP)$$

In this pattern, the first noun phrase is expected to represent the expansion, while the second noun phrase inside the parentheses represents the acronym. To ensure that the first phrase is indeed a valid expansion of the second phrase, we use Algorithm 5.1 to generate a regular expression pattern from the acronym, which is then matched to the expansion term. If multiple noun phrases exist that match the same acronym, we select the shortest valid noun phrase.
After extracting all candidate phrases from the context, we select those candidates where the acronym matches the label of an attribute in the table. The selected candidates are then used to annotate the respective attribute, similar to qualifying noun phrases.

5.4.3 Indirectly Related Context

Direct matching of attribute labels to context phrases, as described in the previous section, enables the extraction of useful annotations, provided the original attribute label represents a good description of the attribute's meaning. However, as we pointed out in Section 3.1, Web tables frequently contain non-informative or empty attribute labels. In addition, we often encounter linguistic variations between the table and the context, where different words are used to refer to the same concept. Consequently, direct matching alone limits the number and type of annotations we can extract from the context. An example, where direct matching fails to extract related phrases, is illustrated in Figure 5.9(c). The generic attribute label *Name* does not provide a good description of the entities in the

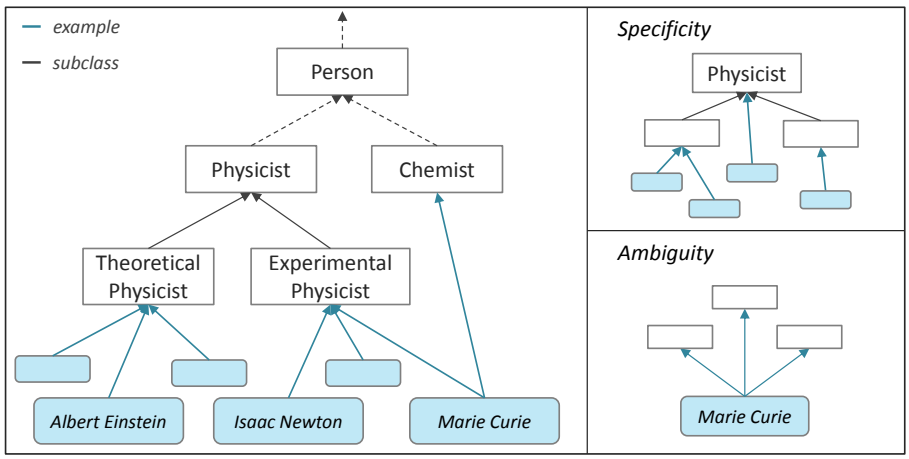


Figure 5.10: Detailed view of an example taxonomy.

column. As a result, we cannot extract the phrases that describe this column.
To address this issue, we consider *indirect matches*, which, in contrast to directly related phrases, are not inferred directly from the attribute label. Instead, we utilize an additional information source to establish a connection between a phrase and a label. In this thesis, we study two different information sources: (1) a *taxonomy* to derive informative labels, and (2) a *linguistic database* to derive linguistic alternatives for the original label.

Taxonomy
A frequently applied technique to recreate attribute labels from a set of instances is to use an external source of commonsense knowledge, such as a taxonomy or ontology, as a reference. A common sense knowledge base usually contains domain-independent information on entities that exist in the real world, as well as their types (i.e. classes or concepts) and the relationships between these types. In the literature, various so-called *conceptualization* approaches have been proposed to infer suitable labels from these knowledge sources. For a review of these techniques see Section 3.3.2.
To infer informative labels for attributes in a table, we consider a knowledge base that stores two types of relationships between instances and classes: (1) a relationship $example(I, C)$ that expresses that an instance I is an example of a class C, and (2) a relationship $subclass(C_i, C_j)$ that expresses that a class C_i is a subclass of another class C_j. Figure 5.10 shows an example of such a taxonomy, where blue boxes represent instances and white boxes represent classes.
Utilizing such a taxonomy, our inference and matching algorithm proceeds as follows:

1. **Extract Candidate Labels:** In a first step, we match attribute instances in the table to entries in the knowledge base in order to retrieve potential class labels. For each instance, we extract multiple candidates (if available).

2. **Aggregate Labels:** In order to retrieve candidate labels that best describe *all* attribute instances, we aggregate the labels retrieved per instance and select the *top k* labels.

3. **Match to Context:** After inferring alternative class labels, we use them instead of the original label to retrieve qualifying noun phrases from the context. The matching process is similar to the one used for direct matching.

Notable Physicists
A physicist is a scientist who does research in physics. Physicists study a wide range of physical phenomena in many branches of physics spanning all length scales ...

Name	Year	Publication
Albert Einstein	1905	„On the Electrodynamics of Moving Bodies “
Isaac Newton	1707	„Arithmetica Universalis“
Marie Curie	1935	„Radioactivity“
...	...	...

Figure 5.11: Example table and context.

Our overall goal is to identify relevant phrases in the context of a table. Therefore, the attribute labels we infer from the taxonomy should be specific enough to avoid matching unrelated phrases. However, if the inferred labels are too specific, we might not find matching phrases in the context. Consider, for example, the classes in Figure 5.10 and the example table in Figure 5.11. While the class *Person* is too generic to describe the scientists listed in the table, the class *Experimental Physicist* is too specific and, thus, less likely to match phrases in the context. To address this challenge, we define two measures, *Ambiguity* and *Specificity*, to evaluate the suitability of classes in the taxonomy. Note that these measures are independent of the instances in the table and, thus, can be precomputed. The measures are defined as follows:

Definition 5.4.1 (Ambiguity)**.** The ambiguity measure addresses the fact that an instance in a taxonomy can be an example of several types, as highlighted in Figure 5.10. We consider a *type* to be a class that has a direct link to the instance. For example, *Marie Curie* has the types *Chemist* as well as *Experimental Physicist*. While, in reality, some types are more plausible than others, we consider all types equally plausible, due to a lack of evidence that suggests otherwise. In probabilistic knowledge bases, relationships between entries are annotated with a plausibility score (Wu et al., 2012).
The ambiguity score $amb(I, C)$ for an instance I and a class C is then calculated as the fraction of types of that instance that are a subclass of class C. For example, $amb(MarieCurie, Physicist) = \frac{1}{2}$, since only one of the two types (i.e *Chemist* and *Experimental Physicist*) of the instance is a subclass of *Physicist*. In contrast, $amb(MarieCurie, Person) = 1$. The ambiguity score reflects the *certainty* of the relation, as it assigns a lower score, if there are several alternative class labels on the same level in the hierarchy, and the highest score $(= 1)$ if there is only one potential class label at that level.

Definition 5.4.2 (Specificity)**.** The specificity of a class C is determined by the number of instances in the taxonomy that directly or transitively belong to this class. The score $spec(C)$ is calculated using Equation 5.19, where $\mathcal{I}$ is the total number of instances in the taxonomy and $\mathcal{I}_C$ is the number of instances that belong to class C:

$$spec(C) = \ln \frac{\mathcal{I}}{\mathcal{I}_C} \tag{5.19}$$

This score is directly inspired by the notion of *IDF* (inverse document frequency), which is frequently used in information retrieval tasks to evaluate term significance. IDF assigns a higher score to terms that appear in only a few documents in a collection, compared to terms that appear in all documents. Similarly, we assign a higher score to classes that cover only a small subset of all instances, and, thus,

are more specific, and a lower score to classes that cover all instances. Consequently, this score keeps generic classes, such as *Thing*, from receiving the highest scores, as we are more interested in specific labels. Recalling the example in Figure 5.10, class *Physicist* receives a higher score than the generic class *Person*.

In summary, the ambiguity score determines how specific a class label is with respect to a specific instance, whereas the specificity score determines how specific a class label is in general (within the taxonomy). Both scores are contrary, as ambiguity favors generic classes, while specificity favors specific classes. In combination, they complement each other to determine suitable candidate labels that can be matched to the table context. The candidate labels for an attribute A in a table are identified using the following successive labeling process:

1. First, for every distinct instance I_A of attribute A, we identify matching instances I_T in the taxonomy. To account for different surface forms due to spelling variations or abbreviations, we use a word-by-word Levenshtein distance to compute the similarity $sim(I_A, I_T)$ between the instance terms. For the subsequent processing steps, we select the *top n* matching instances in the taxonomy for each attribute instance.

2. For each matching instance I_T, we retrieve all classes that the instance is a member of. We consider a class label C, if there is a path of length $l \leq l_{max}$ between I_T and C in the taxonomy.

3. For each retrieved pair (I_T, C), we compute a score using Equation 5.20, which takes the ambiguity as well as the specificity into account.

$$s(I_T, C) = amb(I_T, C) \cdot spec(C) \tag{5.20}$$

4. For each distinct attribute instance I_A, we retrieve the best score per class C using Equation 5.21

$$s(I_A, C) = \max_{j \in \{1,\ldots,n\}} (sim(I_A, I_{T_j}) \cdot s(I_{T_j}, C)) \tag{5.21}$$

5. After retrieving scored class labels for each distinct attribute instance, we need to aggregate the scores across instances to determine the label that best described the attribute. The aggregation formula is shown in Equation 5.22, where $\mathcal{I}_A$ is the set of distinct instances of attribute A. Finally, the *top k* class labels with the highest overall scores are selected as suitable attribute labels that can be matched to phrases in the table context.

$$s_A(C) = \frac{\sum_{I_A \in \mathcal{I}_A} s(I_A, C)}{|\mathcal{I}_A|} \tag{5.22}$$

Depending on the coverage of the knowledge base utilized for inference, we can identify suitable labels for a wide range of attributes. However, in many cases, this approach is limited to attributes that contain named entities, such as cities, countries, or people and places of public interest. To address this limitation, there is ongoing research directed at automatically extracting commonsense knowledge from the Web in order to achieve greater coverage (Wu et al., 2012). Still, it is unlikely to find matches for all entries of a table in such a knowledge base.

Linguistic Database
A complementary approach to discover indirect matches in the context is the utilization of a *linguistic database*, such as WordNet (Miller, 1995). The previous matching techniques do not take a potential vocabulary mismatch between the attribute label and the contextual description into account. Consequently, if the original label or the label derived from a taxonomy differ from the terms used in the context, we could not establish a match, so far.
Consider, for instance, the example in Figure 5.9(c). For the attribute initially labeled *Name*, we can derive the alternative label *Garrison* from a taxonomy, based on the entries in the column. However, there are no mentions of the word *Garrison* in the context, and, instead, the word *Fort* is used. Such a mismatch is common in many text-based applications, as most human languages are very versatile and expressive.
We address this issue by deriving linguistic alternatives of the attribute label from a linguistic database, similar to query expansion in Web search applications. As alternative labels, we focus predominantly on synonyms. Hypernyms and hyponyms can also be used. However, the selection of suitable candidates from the set of available hypernyms and hyponyms is subject to future work. The retrieved labels can then be matched to the context, again using the technique outlined in Section 5.4.2 to extract relevant noun phrases.

5.5 EXPERIMENTAL EVALUATION

In this section, we evaluate the context extraction algorithm proposed in the previous section. For each extraction technique, we provide a qualitative and quantitative analysis of the extracted information. We also study the impact of context relevance on the extraction quality. Finally, we show the benefit of column-specific context information for the understanding of tables in a table search application.

5.5.1 Experimental Setup

Before presenting the results of our evaluation, we specify the dataset and parameters settings used in the experiments.

Dataset
For this evaluation, we extracted 850 tables from the English Wikipedia, focusing mainly on tables with non-informative attribute labels like *name* or *title*, or generalized labels like *location*. For each table, we extracted context information from the corresponding Wikipedia page, including any available caption, the closest section headline, and the text of the surrounding section. In addition, we extracted the full text of the Wikipedia article to use as a baseline and to extract relevant context sections. Table 5.4 shows some statistics of the extracted context information.

Context Relevance Estimation
Using the process described in Section 5.3, for each table, we extracted relevant context sections from the Wikipedia page. For the linear text segmentation, we applied the sentence-based TextTiling approach with block size $b = 6$ and depth score threshold parameter $t = 0.5$. After splitting the context

Table 5.4: Statistics of context extracted for tables used in evaluation.

Context	# of Tables	Avg. Word Count
• Caption	43	4.72
• Headline	406	2.13
• Surrounding Text	770	228.03
• Relevant Text	848	459.20
• Full Text	850	1,599.99

into coherent segments, we used the cosine similarity of TF weighted term vectors to estimate the relevance of each segment. For each table, we then retrieved the top 3 segments as relevant context.

Noun Phrase Extraction For each table, we applied the extraction algorithm proposed in Section 5.4 to retrieve attribute-specific information from the various context sections. To identify noun phrases in the context, we utilized the Stanford Parser (Socher et al., 2013). For the inference of alternative column labels, we use the *Simple Taxonomy* in *YAGO* (Suchanek et al., 2007), a publicly available knowledge base based on facts extracted from Wikipedia and other Web sources. For the scoring function, we set the parameters $l_{max} = 5$, $n = 1$ and $k = 3$ (see Section 5.4.3). To retrieve synonyms of attribute labels, we utilize WordNet (Miller, 1995) as the linguistic database.

5.5.2 Column-specific Context

In our first set of experiments, we study the quality and quantity of the phrases extracted from the context. We differentiate between the various context section, i.e. caption, headline or text, as well as the different matching techniques proposed in Section 5.4.

Annotation Statistics
The 850 tables in our test set have an average number of 4.15 attributes per table. With our extraction algorithm, we were able to extract phrases from the context for an average of 2.23 attributes, i.e. roughly half of all attributes, per table. This number reflects the fact that generally not all attributes of a table are explicitly mentioned in the context.
In total, we extracted 11,857 potentially related noun phrases from the context and 4,252 alternative attribute labels from YAGO. Table 5.5(a) shows the distribution of extracted noun phrases across the various context sections. As expected, more phrases were extracted from longer context sections, with only a few phrases extracted from the captions. However, only 5% of all tables had a `<CAPTION>` tag to begin with. Considering only relevant context, instead of all context, reduced the number of extracted phrases by more than 50%. However, compared to using only the closest paragraph, the relevant context selected by our algorithm returned significantly more noun phrases.
Table 5.5(b) shows the number of extracted phrases for each matching technique. In the table, we refer to each technique by the phrases that were used for matching. The attribute label is the label originally used in the table, while the YAGO label is inferred from the knowledge base. *Syn* indicates

Table 5.5: Related noun phrases (NP) per context section and per matching technique.

(a)

Context	# of NPs
• Caption	8
• Headline	92
• Surrounding Text	1,513
• Relevant Text	5,143
• Full Text	11,857

(b)

Technique	# of NPs
• Attribute Label	3,457
• Attribute Label Syn.	1,173
• YAGO Label	4,442
• YAGO Label Syn.	2,785
• Acronyms	2
• All	11,857

that a synonym of the respective label was used. The table shows that most phrases were identified by matching YAGO labels to phrases in the context. This is due to the fact that we have a large number of tables with non-informative labels, such as *name* or *title*, which are less likely to appear directly in the context. Acronyms are less frequent in attribute labels. Consequently, we have significantly less phrases extracted using acronyms.

Annotation Quality

Manually, evaluating all extracted phrases is a very extensive and time-consuming task. Therefore, we only evaluate a sample of the phrases to provide an indication of the quality and accuracy of the phrases. For each type of context and each matching technique, we drew a random sample of 25 noun phrases extracted from the context. Using human judges, we evaluated the correctness and relevance of each phrase with respect to the associated attribute, assigning a score of 1 if the phrase offers a correct description of the attribute, 0.5 if the phrase is related, but not an exact description, and 0 if the phrase is not related to the attribute. The average scores are shown in Figure 5.12.

The results indicate that the closer a phrase is to the table on the Web page, the more likely it is indeed related to an attribute in the table. While phrases extracted from the captions or closest section headlines are of high quality, the overall label quality is much lower, indicating more noise in the context. However, we can see an improvement in the label quality when considering only relevant context instead of all context. The relatively low score for the surrounding text is in part due to the fact that the closest paragraph is not necessarily topically related and, thus, can also introduce noise.

Considering the different matching techniques, we can see that matches based on synonyms are more likely to retrieve incorrect noun phrases. This is in part due to the synonym expansion technique we employ. A more conservative approach, which puts more limits on the selection of meaningful synonyms, is expected to retrieve less unrelated phrases.

Overall, our heuristic context extraction algorithm retrieves many meaningful phrases to extend the attribute descriptions in Web tables, but also retrieves many inaccurate phrases, especially if too much noisy context is considered. However, many table processing application benefit from the attribute-specific context information. In the next section, we evaluate this benefit, using a basic search application as an example.

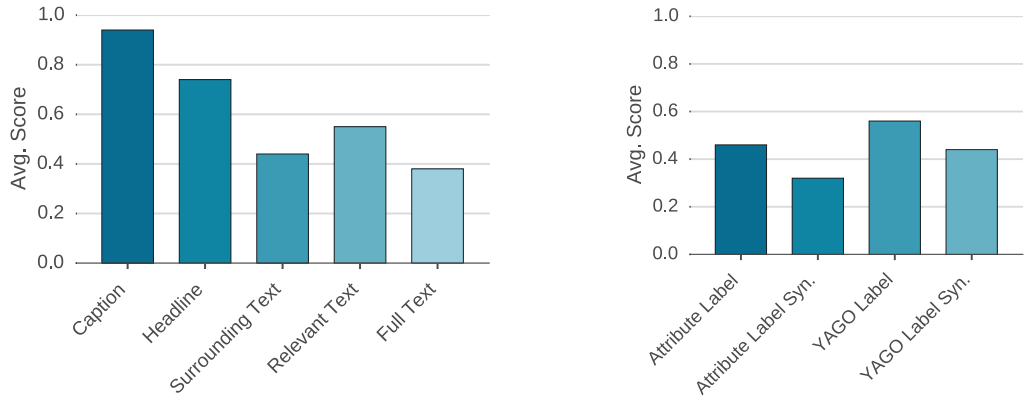


Figure 5.12: Estimation of the quality of extracted noun phrases per context section and per matching technique.

5.5.3 Context in Search Applications

Similar to document retrieval, we can search for relevant information in tables using simple key word queries. However, tables alone often do not provide enough explicit information to give a description of the table content that is sufficient for effective retrieval. Extending the table description with content information provides a richer description, but also introduces noise that can mislead the retrieval algorithm. As Pimplikar et al. (2012) pointed out, a decision about the relevance of a table to an information need should not be made on the context alone, but in conjunction with the attributes in the table. A matching keyword in the context does not guarantee an implicit match with table content, since the context can be very unspecific.
Our hypothesis is that the ranking of relevant tables in a keyword search can be improved with attribute-specific context annotations by placing more emphasis on context information that can be associated with a specific attribute in the table. Furthermore, mapping query terms to attributes can benefit from the richer description of the attributes provided by the annotations. To validate this hypothesis, we consider a simple search application that retrieves individual attributes as well as relevant tables based on keyword queries, and evaluate the retrieval quality on real-world Web tables.

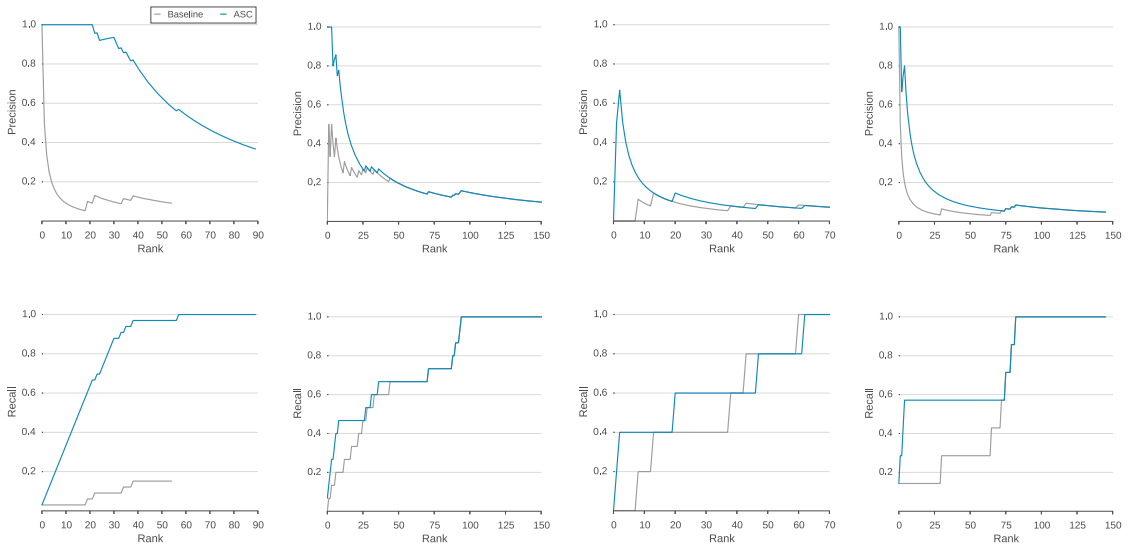
Setup
For the search application, we utilize the open-source search library *Lucene*[1] to index all tables and context sections. Lucene provides standard retrieval functionality for documents, offering the separate indexing of different document sections for a faceted search. In our search scenarios, we differentiate between information found in the header of a table, the extracted attribute-specific annotations as well as the unprocessed context in general. In Lucene, we can assign different *boost factors* to indicate and influence the relevance of these different facets. We assign a boost factor of 2 to header information, 1.5 to the extracted annotations and 1 to the full context.

[1] `https://lucene.apache.org`

To score the indexed documents using Lucene's adapted version of the cosine similarity of TF-IDF weighted term vectors, which incorporates the various boost factors. For the search, we focus on single attribute keyword queries as described by Pimplikar et al. (2012). These queries target a single attribute in a table, predominantly to retrieve the names of entities. Example queries include "*prime ministers of England*" or "*Australian cities*". We expect these queries to benefit from the attribute-specific annotation we extracted from the context.
To evaluate the retrieval performance, we use the standard metrics *precision* and *recall*. For retrieval systems, precision measures the percentage of retrieved documents that are relevant to the query, while recall measures the percentage of all relevant documents that were retrieved with respect to all relevant documents in the index.

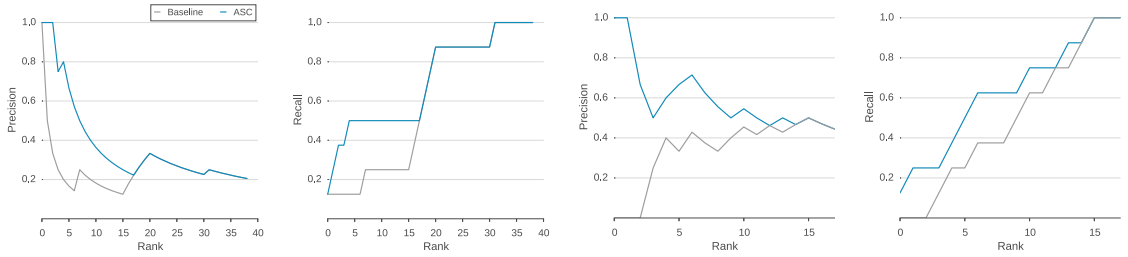
Attribute Search
For the attribute search, we treat each attribute in a table as an individual Lucene document and index, if available, the original attribute label, the attribute-specific annotations as well as the complete context, each in a separate facet. We selected a set of keyword queries such that a search on the attribute label alone would not return any results. For these queries, additional context information is required to identify relevant attributes. To measure precision and recall, we ran each query on all indexed documents and manually labeled documents in the result sets as relevant or irrelevant.
In Figure 5.13, we show the precision and recall for the ranked results of four selected queries: *"football player"*, *"video games"*, *"short films"* and *"record labels"*. The blue lines correspond to the results achieved when indexing attribute-specific annotations in addition to header and general context information. The grey lines represent the baseline, which only takes header information and general context into account. For each query, we can see that attribute-specific annotations improve the ranking, resulting in higher precision and recall for the top-ranked columns.
The query *"football player"* shows the most significant improvement over the baseline. Here we retrieved more relevant attributes via annotations than we could retrieve using only the context. This is due to the fact that for many attributes we could infer the label *football player* directly from YAGO, while the context provided only very little information. Thus, combining annotations retrieved from the context with annotations inferred from an external knowledge base further improves the ranking of attributes in table retrieval.

Table Search
Not only the search for specific attributes can benefit from the annotation, but also table search in general. Although less specific than the search for individual attributes, context information is more likely to be relevant for a table, if it can be associated with an attribute in the table. This is directly related to statement by Pimplikar et al. (2012), that the relevance of a table with respect to a query should be evaluated in conjunction with attributes in the table and not solely based on the context. Establishing an association between phrases in the context and attributes in the table avoids matching tables where the search term is only mentioned in the context, but is not related to the table.
To see if table search can benefit from attribute-specific context annotations, we index all tables, this time treating the entire table as a single document. Again, we index all attribute labels, the annotations as well as the complete context in separate facets and assign boost factors. For the baseline, we only consider the attribute labels as well as the unprocessed context.



(a) "football players" (b) "video games" (c) "short films" (d) "record labels"

Figure 5.13: Precision and recall for selected column search queries.



(a) "record labels"' (b) "short films"

Figure 5.14: Precision and recall for selected table search queries.

To evaluate the performance, we ran the same queries as before, manually judging the relevance of the retrieved tables. We regard a table as relevant to the query, if at least one attribute in the table contains the desired entities.

In Figure 5.14, we show the precision and recall of the ranked tables for the queries "*short films*" and "*record labels*". Again, we can see that the attribute-specific context annotations lead to a higher rank for relevant tables, which improves precision and recall.

Overall the results of the attribute and table search experiments indicate that our hypothesis is valid. Attribute-specific context information can improve table retrieval by providing a richer description for attributes and emphasizing contextual information that is directly associated with the table content.

5.6 SUMMARY AND DISCUSSION

On the Web, context in the form of headlines, captions or surrounding text often provides supplementary information that is important for the correct inference of the meaning and intention of a Web table. It frequently offers a detailed description of attributes, indicates restrictions or contains hidden attributes. As the representation of information in tables is generally sparse and often implicit, in many cases with non-informative or generic attribute descriptions, it is difficult to understand tables in isolation. As a result, various table processing tasks benefit from the additional information in the table context, as it can be used to recover the missing information.
However, not all information mentioned in the context is necessarily relevant to the table and, especially when considering large texts, the verbosity of the context can lead to incorrect interpretations. In the literature, the relevance of Web table context has only received very little attention so far, with many researchers simply considering either the entire available context, or no context at all. If context information is incorporated, it is mostly regarded as a general indicator for the table content and meaning, but rarely to extract specific details, largely due to the noise in the text.
In this chapter, we extended the Web table understanding process to incorporate contextual information in a more focused fashion. First, we addressed the availability and relevance of contextual information for Web tables, then utilized the context to extract attribute-specific information to describe the table content. In summary, we made the following contributions:

- **Context Characteristics and Availability:** First, we studied the different resources for context information that are available for tables embedded in HTML pages as well as tables on designated data platforms. For each resource type, we identified characteristic features that impact the processing of the context.
- **Context Relevance Survey:** In an initial user study, we estimated the relevance of different context resources with respect to the table content, and confirmed the assumption voiced in the literature that large text segments are the least specific context resource and most likely to introduce noisy information.
- **Estimating Context Relevance:** We addressed the issue of noise in large context segments by proposing a paragraph selection approach to identify relevant paragraphs in the context. The algorithm relies on word-based similarity measures to estimate the relevance of individual context segments before selecting only the most relevant paragraphs based on an adjustable threshold.
- **Attribute-Specific Context:** After evaluating the relevance of Web table context, we utilized the context to extract attribute-specific information to provide a richer description of the table content. We proposed an extraction algorithm that performs direct as well as indirect matching of attribute labels to selected phrases in the context.
- **Experimental Evaluation on Real-World Data:** Finally, we conducted various experiments, including a search application, on real Web tables in order to analyze the impact of context relevance and to highlight the benefits of attribute-specific context information.

Our experiments confirm the initial assumption that the content of a table is often not sufficient to describe its complete meaning and additional information from the table context is required to "understand" a table algorithmically. Applications such as table search benefit from the supplementary

information to produce a better ranking of potentially relevant tables. Furthermore, considering the general relevance of the context, as well as establishing an explicit connection between phrases in the context and attributes in the table further reduce noise and improve the quality of table-based applications.

While the algorithms we proposed achieve good results for a wide range of tables, there is still room for improvement regarding the accuracy of the extracted context information. However, to further increase the precision of the extraction algorithm, a more detailed syntactic and semantic analysis of the context is required. Deep semantic analysis is both challenging and computationally expensive. Before incorporating such complex algorithms into the table recovery process, it is necessary to assess whether the benefits justify the additional costs.

6

SEMANTIC NORMALIZATION

6.1 Relation to Database Normalization

6.2 Formalization of Semantic Normalization

6.3 Challenges

6.4 Normalization Process

6.5 Experimental Evaluation

6.6 Summary and Discussion

To fully utilize the data stored in tables on the Web, we must be able to *understand* their content. However, without a conceptual model or formal description, we can only attempt to infer the semantics directly from the table, which is a very challenging task. In order to limit the complexity of this inference, the majority of previous research addressing the interpretation and analysis of tables on the Web, has focused exclusively on tables that only describe a single semantic concept. We refer to these tables as *single-concept tables*. While some studies specifically exclude other types of tables from their consideration, others simply assume that *all* tables on the Web are single-concept tables. This *single-concept assumption* is largely motivated by the relatively small size of Web tables (see Section 3.1 for statistics). In the context of Web table understanding, single-concept tables are frequently assumed for the extraction of binary relations from tables (Yakout et al., 2012; Zhang et al., 2013), matching tables to concepts in an external knowledge base (Venetis et al., 2011; J. Wang et al., 2012), as well as finding related tables (Das Sarma et al., 2012).
However, the Web also provides access to a substantial amount of larger, more complex tables. In addition to Web pages, we can find larger tables especially on public Open Data portals, which do not impose size restrictions on the tables in the same way as Web pages. Consequently, we also encounter many *multi-concept tables*, i.e. tables that describe properties of multiple concepts as well as the relationships between them. Figure 6.1 shows an example of such a table, which mentions the semantic concepts *City*, *Country* and *Mayor*, with additional properties for the first two.

City	Mayor	Elevation(m)	Country	Area (km²)	Population (Mio.)
London	B. Johnson	35	England	130,395	53
Manchester	N. u. Hassan	38	England	130,395	53
Dublin	O. Quinn	-	Ireland	70,273	4.59
Berlin	K. Wowereit	34	Germany	357,021	80.5
Paris	B. Delanoë	35	France	551,695	63.4

City

Name	Mayor	Elevation(m)	Country
London	B. Johnson	35	England
Manchester	N. u. Hassan	38	England
Dublin	O. Quinn	-	Ireland
Berlin	K. Wowereit	34	Germany
Paris	B. Delanoë	35	France

Country

Name	Area (km²)	Population (Mio.)
England	130,395	53
Ireland	70,273	4.59
Germany	357,021	80.5
France	551,695	63.4

Figure 6.1: Example of a table describing three separate semantic concepts: the concept *City* with additional property *Elevation*, the concept *Mayor*, and the concept *Country* with properties *Area* and *Population*.

Treating such a table as a single-concept table, i.e. assuming that all attributes describe the *same* semantic concept, leads to an incorrect interpretation and, ultimately, use of the data. To illustrate these repercussions, we take the extraction of binary relations as an example. Decomposing tables into binary relations is an integral part of the *Infogather* system developed by Yakout et al. (2012). The binary relations of interest are those that combine an attribute containing entities with an attribute describing a property of these entities, such as the relation (*City*, *Elevation*). Assuming a single-concept table

with n attributes and a simple key, we can extract $n - 1$ binary relations, combining the key attribute with each of the remaining attributes. In our example, we would extract (*City*, *Area*) and (*City*, *Population*) as two of these relations. However, it is clear to see from the data that these are not valid relations reflected by the table. Instead, the correct relations are (*Country*, *Area*) and (*Country*, *Population*). To prevent such incorrect data from being extracted from Web tables, we need to identify the individual concepts and their associated attributes.

In this chapter, we address the limitations and consequences of a single-concept assumption in the context of Web table understanding. We propose a *semantic normalization* approach, which identifies concept boundaries in wide, multi-concept tables. Based on these boundaries, the tables can be decomposed into multiple single-concept tables, which can then be processed as before. Our approach serves as a requisite preprocessing step for many Web table understanding and integration techniques, in order to process multi-concept tables correctly.

Before studying the task of semantic normalization, it is important to note that normalization is not equally relevant for all classes of tables encountered on the Web. Recalling the classification introduced in Section 3.1, only tables describing multiple different attributes are affected by the implications of a single-concept assumption. Consequently, the focus of this chapter is mainly on entity-attribute tables and their transposed counterparts. Matrix tables and other tables that only describe a single entity that is factored out of the table, i.e. whose name or reference is only mentioned in the table context, are unlikely to describe multiple concepts.

The notion of semantic normalization is closely related to traditional normalization techniques in relational databases. Therefore, we first review related work in this area of research (Section 6.1). After that, we formally define the objective of semantic normalization, as well as related concepts (Section 6.2). Following the problem statement, we review characteristic features of Web tables that introduce unique challenges for the normalization task (Section 6.3). We then introduce our solution for the semantic normalization of Web tables with a detailed specification of the processing steps involved (Section 6.4). We also propose an extension to the basic normalization process in order to further optimize the result quality (Section 6.4.5). The conceptual specification of the algorithms and workflow involved in our approach is followed by an experimental evaluation on real-world data (Section 6.5). We then conclude this chapter with a summary and discussion of our contributions (Section 6.6).

6.1 RELATION TO DATABASE NORMALIZATION

Semantic normalization and database normalization are closely related concepts, as both aim at logically grouping attributes in a relation based on semantic constraints. However, both tasks have different motivations and, as a result, different expectations regarding the target schema. In the following sections, we first review the foundations of database normalization. Subsequently, we review previous research in different areas related to database normalization and evaluate, if any of the established techniques can be applied to the semantic normalization of Web tables.

6.1.1 Foundations of Database Normalization

In many application scenarios of relational database management systems, *normalized* tables are a desired goal in the database design process. *Database normalization*, which was first formalized by E. F. Codd (1971), groups attributes of a relation schema logically, so that only related data is stored in the same table. The purpose and main motivation for normalizing a database is to minimize redundancy in the data and avoid inconsistencies and modification anomalies (Abiteboul et al., 1995). Relational database design, including the specification of normal forms and the generation of normalized schemas, has been studied intensively in the 1970s and 1980s.

Normal Forms
The level of normalization of a relational schema is specified through a set of *normal forms*, which are defined on top of each other. A normal form defines the set of dependencies that are allowed to hold (Abiteboul et al., 1995). As a result, each normal form provides different quality and consistency guarantees.
Established normal forms include First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF) and Boyce-Codd Normal Form (BCNF). Higher order as well as alternative normal forms exist, although they are not as commonly used. Each of the common normal forms is defined on top of the previous level, further restricting the set of valid dependencies, with BCNF defined on top of 3NF. In database terminology, a schema is considered to be *normalized*, if it is, at least, in 3NF or BCNF. Normal forms up to BCNF are defined on the central concept of *functional dependencies*, except for 1NF, which simply requires all attribute values to be atomic.

Functional Dependencies
Functional dependencies (FDs) reflect a semantic constraint between two sets of attributes in a relation. In the design process of a relational database, they are derived from the application logic or specified by a domain expert, and ensure the integrity of the data. Formally, we can define functional dependencies as follows:

Definition 6.1.1 (Functional Dependency)**.** Let R be a relation schema with $X \subseteq R$ and $A \in R$. Then a relation r over R satisfies the *functional dependency* $X \rightarrow A$, if for every pair of tuples $t, u \in r$ holds: $t[B] = u[B]$ for all $B \in X \Longrightarrow t[A] = u[A]$. Simply, if t and u share a value for X, they must also share a value for A. X is said to be the *determinant* and A the *dependent* of the functional dependency.

In addition to this general definition, we distinguish between different characteristics of functional dependencies. A *trivial* FD is a dependency $X \rightarrow A$, where $A \subset X$. A functional dependency is called a *simple* dependency, if the determinant is a single attribute. Furthermore, a functional dependency is considered a *full* FD, if no functional dependency $Y \rightarrow A$ exists, with $Y \subset X$. Otherwise the dependency is considered to be *partial*.
On top of this general specification of functional dependencies, a set of properties and axioms have been defined, most prominently Armstrong's axioms, which include reflexivity, augmentation and transitivity (Armstrong, 1974).
These properties and inference rules allow for the specification of two features that are central to various normalization algorithms: the *closure* and the *minimal cover* of a set of functional dependencies.

The closure F^+ is the set of all functional dependencies that can be derived from the initial set F by applying rules or axioms. A set of dependencies G is a cover of F, if all functional dependencies in F can be derived from G, such that $F^+ \subseteq G^+$. Such a cover is *minimal*, if there is no proper subset $G' \subseteq G$ that is also a cover of F.

Normalization Process
Given a universal relation and a set of functional dependencies that hold in the data, two main techniques have been proposed to generate a normalized schema. The first approach considers normalization as a form of *decomposition* of the source schema, identifying dependencies that violate the requirements of a normal form and stepwise splitting the relations accordingly, until all requirements of a specific normal form are met. The second approach considers the *synthesis* of a target schema that conforms to a normal form from the set of dependencies. Simple relations are formed based on the minimal cover of the dependencies.
Decomposition and synthesis can result in different schemas, with different properties. Two of the most important properties that are considered in normalization are a *lossless decomposition* and *dependency preservation*. A normalization is considered lossless, if the source schema can be reproduced from the target schema through natural joins of relations. Also, the dependencies are preserved, if all functional dependencies that hold in the source relation still hold in the target relation. In many applications, an algorithm that guarantees both properties is favored.

6.1.2 Related Work

Database normalization and the formal guarantees it provides with respect to data consistency and dependency are an integral part of database design and other related tasks. As a result, aspects related to the normalization of databases have been studied extensively in the literature.

Automating Database Normalization and Schema Design
Database normalization is traditionally a manual design process that requires domain knowledge and experience to specify the functional dependencies in the data. Especially for large databases that describe numerous concepts and relationships, normalization is a complex task. As a result, research has focused on automating database normalization, which has been investigated for many years, with interest in the field re-occurring on a regular basis. In the literature, we can distinguish two main directions of research: (1) the inference of keys and dependencies from the data, and (2) the actual *normalization*, i.e. decomposing or synthesizing relations based on functional dependencies in order for the schema to comply to a certain normal form.
The inference of functional dependencies is the central challenge of database normalization. In addition to identifying a set of functional dependencies that hold in the data (Mannila et al., 1994), deriving the closure of the set as well as the minimal cover are two key issues (Huhtala et al., 1999). As relation and schema sizes grow, FD inference becomes increasingly expensive and time consuming, as documented by Mannila et al. in their analysis of the problem's complexity (Mannila et al., 1992). For a fixed set of functional dependencies, they measure the time complexity of inference via a brute force approach as $O(n^2 2^n p \log p)$, where n is the number of attributes and p the number of tuples in a relation. To address this challenge, several efficient and scalable inference algorithms have been developed, including *TANE* (Huhtala et al., 1999) and *FUN* (Novelli et al., 2001), which utilize

efficient data structures as well as sophisticated pruning techniques the reduce the computational costs. The *TANE* algorithm, for instance, reduces the time complexity of the dependency inference to $O(s(p + n^2) + kn^3)$, where k is the number of keys in the relation and s is the number of partitions formed by the algorithm, which has a worst case of $O(2^n)$, but is often significantly lower in practice. The algorithm achieves a linear dependency on the number of tuples p, compared to the $\Omega(p \log p)$ dependency reported by Mannila et al., which significantly reduces the computational costs.
In general, all functional dependencies inferred from the data are considered to be correct. This assumption is justified, if the size of the sample, i.e. the number of tuples, considered in the inference is large enough to form a realistic representation of the domains of the attributes over which the dependencies are defined. However, for public tables on the Web, which often contain only a couple of rows, this is generally not the case. As a result, a direct application of established inference algorithms to Web tables often results in low precision.
Normalization itself has also been studied extensively, both conceptually as well as practically. Conceptually, two general approaches have been proposed, both controlled by a set of functional dependencies. Decomposition was first proposed by E. F. Codd (1971), with subsequent contributions from Delobel et al. (1973) as well as Tsou et al. (1982), among others. Examples for synthesis approaches include the work of Bernstein (1976) and the techniques of C. P. Wang et al. (1975). In addition to established normal forms, more arbitrary forms of decomposition have been studied as well, including binary decomposition (Dechter, 1987). In order to automate the process of database normalization, practical aspects including efficient data structures (Bahmani et al., 2008) as well as paralell execution (Bahmani et al., 2010), have been studied, as well.

Reverse Engineering of Relational Databases
Another related area of research is *reverse engineering* of relational databases, which refers to the analysis and interpretation of an existing database in order to identify the conceptual structure underlying the data (Chiang et al., 1994). The conceptual information is then represented at a higher level of abstraction, generally using conceptual models, such as entity-relationship (ER) and extended entity-relationship (EER) models, binary relationship (BR) models or object-modeling technique (OMT) models. Obtaining such a conceptual model of the data is beneficial for a wide range of applications, including resolving semantic degradation in the schema, redesigning a database, as well as integrating heterogeneous sources (Chiang et al., 1994).
Reverse engineering of relational databases has been studied extensively and many advanced methods have been proposed. However, several assumption are frequently made, which limit the applicability of reverse engineering methods to the process of Web table understanding. The first assumption, which forms the basis for the majority of proposed methods, addresses the state of normalization of the source schema. In most cases, including the approach introduced by Chiang et al. (1994), the source schema is assumed to be in 3NF. In addition, reverse engineering of databases is generally designed and implemented as an iterative and interactive process that requires input, intervention or validation from a user or domain expert (Premerlani et al., 1994; Petit et al., 1996). Information frequently provided by the user to specify additional constraints over the data include candidate keys, inclusion dependencies, partial dependencies or many-to-many relationships (Shoval et al., 1993). On top of information supplied by the database itself as well as the user, some methods rely on auxiliary sources, such as index structures and query logs stored in the DBMS in oder to infer the semantics of the database (Petit et al., 1996). Obviously, considering autonomous tables on the Web, which are

not associated with other tables in a database, we do not have access to these resources.
Finally, many of the proposed methods for inferring constraints and relations from a database presume clear semantics and high quality data. Attributes bearing the same name are generally assumed to carry the same meaning, which clearly does not hold with respect to heterogeneous Web tables. Additionally, attribute values, especially in key columns, are assumed to be accurate and consistent (Chiang et al., 1994).
In recent years, many of the techniques proposed for reverse engineering of relational databases have been adopted and extended in the context of the Semantic Web. Instead of deriving a conceptual model, an ontological representation of the data is generated and mapped to existing ontologies (Spanos et al., 2012).

Normalization of Dataspaces
A third area of research that studies the decomposition of relational schemas is the normalization of dataspaces. In contrast to the traditional data management in relational DBMS, which requires all data to be semantically integrated, in dataspace support platforms (DSSP), services, such as search or querying over the data, are provided over a loose collection of disparate sources that are not fully integrated on a semantic level. The schema defining the individual data sources and their interrelations is not determined upfront through an elaborate integration process, but evolving continuously in a *pay-as-you-go* fashion (Franklin et al., 2005). As a result, DSSPs offer data management functionality over heterogeneous sources with significantly less integration effort early on, but provide weaker guarantees with respect to data consistency and durability, until a tighter integration is achieved (Halevy et al., 2006).
In order to provide data management services with only little integration effort for the user, DSSPs rely on automatically generating probabilistic mappings as well as a probabilistic mediated schema to describe the relations between the separate data sources (Das Sarma et al., 2009). As the dataspace grows and the level of integration increases, such a mediated schema also increases in size, making it difficult for users to comprehend the semantics of the data. At this point, normalizing the mediated schema and, thus, decomposing it into smaller relations, is necessary to support the user. In this context, D. Z. Wang et al. introduce the notion of *semantic normalization*. It is described as the decomposition of a large mediated schema into several smaller schemas, by grouping semantically related attributes (D. Z. Wang et al., 2009). Probabilistic functional dependencies are inferred from the data across the various sources and used to drive the normalization process. After an initial pruning step, the inferred functional dependencies are considered deterministic and decomposition into a dependency preserving 3NF is performed.
The objective as well as the implementation proposed by D. Z. Wang et al. are closely related to our goal of semantically normalizing multi-concept tables. However, there are also important differences and limitations that prevent a direct application of the proposed approach to our scenario. First of all, inferring a single mediated schema from millions of Web tables is not feasible. D. Z. Wang et al. consider only sources from a selected domain, while our goal is an open-domain solution (see Section 3.1). Second, inferred functional dependencies (after pruning) are considered to only represent meaningful dependencies. This assumption is reasonable, if sufficient samples are considered, for instance by combining data from multiple sources. However, individual Web tables often do not provide sufficient samples. As a result, standard normalization techniques, which assume all provided functional dependencies to be valid, cannot be applied directly to normalize multi-concept Web ta-

bles. Finally, inferring probabilistic mappings between data sources that are as heterogeneous and ambiguous in their description as Web tables, is a challenging task by itself and is afflicted with a considerable level of uncertainty (Das Sarma et al., 2009). We address these challenges in more detail in the subsequent sections.

6.2 FORMALIZATION OF SEMANTIC NORMALIZATION

Our goal is the identification of individual semantic concepts, more specifically the attribute sets that make up the relational representation of these concepts, in wide universal Web tables that describe more than one concept. After reviewing key concepts of relational data modeling, including semantic constraints, we can now provide a more formal description of semantic normalization in the context of Web tables.

Relational tables, in general, provide information on entities and their relationships in the form of attribute sets. The way in which concepts (i.e. entity types) and relationships are described in a table can be very versatile and complex. However, most Web table applications, such as entity augmentation (Yakout et al., 2012; Zhang et al., 2013) and fact search (J. Wang et al., 2012), focus predominantly on *simple concepts* and *binary relations*.

A simple concept is described, in relational form, by a *simple* key K, i.e. a single attribute (or column) that provides a name or identifier for instances of the concept, and a number of additional attributes A_i, which contain property values (Zhang et al., 2013; Yakout et al., 2012). For example, in Figure 6.1, the attributes *(Name, Area, Population)*, with *Name* as the key, describe the simple concept *Country*.

A binary relation is a relation that holds between *two* attributes. On a conceptual level, we consider two types of binary relations in Web tables. First of all, a binary relation can be an *entity-attribute binary (EAB)* relation (Yakout et al., 2012), which holds between the key and an additional attribute of the same concept. An example is the relation *(City, Elevation)* in Figure 6.1. Second, binary relations can be *entity-entity binary (EEB)* relations that hold between two concepts represented by their keys, for instance the relation(City, Country). In general, binary relations between concepts can be $1:1$, $1:n$ or $n:m$ relations.

Based on these characteristics, we derive the following definitions of single-concept and multi-concept tables at the relational level:

Definition 6.2.1 (Single-Concept Relation)**.** Let R be a relation with attributes $K, A_1, \ldots, A_n$, with simple key K. Then R corresponds to a *single-concept* relation R_C, if every $A_i \in R$ is a valid attribute of the same semantic concept C, and there is a non-transitive functional dependency $K \rightarrow A_i$ (forming a binary relation).

Definition 6.2.2 (Multi-Concept Relation)**.** A relation R corresponds to a *multi-concept* relation, if it can be described as the result of a natural join (or multiple joins) between several single-concept relations R_{C_i}. For every concept C_i, R contains a simple key K_i (i.e. an entity column). For each attribute $A_i \in R, A_i \neq K_i$, there is exactly one concept C_i so that there is a non-transitive functional dependency $K_i \rightarrow A_i$ between the concept's key and the attribute.

Based on these definitions and following the specification by D. Z. Wang et al. (2009), *semantic normalization* can then be defined as follows:

Definition 6.2.3 (Semantic Normalization)**.** The objective of *semantic normalization* is the detection and separation of individual semantic concepts C_i contained in a relation R. Multi-concept relations are split into multiple single-concept relations $R_{C_i} \subset R$, one for each concept. Single-concept relations remain unchanged.

Both, semantic normalization and database normalization, cluster semantically related attributes. However, instead of minimizing redundancy in the data and avoiding modification anomalies, semantic normalization aims at identifying the conceptual structure within a large relation and deriving a schema that better reflects this structure, in order to prevent incorrect interpretation and information extraction.
Directly applying standard normalization techniques to achieve semantic normalization (in the context of Web tables) faces several issues. First of all, not all normal forms considered in database normalization guarantee to preserve concept boundaries (D. Z. Wang et al., 2009). Furthermore, these techniques require a definite set of valid functional dependencies, which drive the normalization process. However, due to the characteristics of tables on the Web, we can only approximate these dependencies.
Consequently, we propose a heuristic process to achieve *semantic normalization*, which identifies concept boundaries and decomposes multi-concept tables into multiple single-concept tables without loss of information.

6.3 CHALLENGES

Public tables on the Web pose unique challenges for the identification of meaningful binary relations between attributes. In general, we do not have a domain expert or the application logic to provide us with functional dependencies. Therefore, we need to recover the dependencies from the tables. To infer *meaningful* functional dependencies, tables provide two important resources that need to be taken into account: the header of the table, which provides a conceptual description of the attributes, and the body, which contains the tuples for which the functional dependencies must hold. In general, verifying that a functional dependency holds in the data is straightforward, due to the precise specification in Definition 6.1.1. However, inferring dependencies from the data is more challenging. In the context of the Web, where the data is very heterogeneous and noisy, we face the following challenges:

1. **Missed Functional Dependencies:** Due to missing and inconsistent information, relevant functional dependencies cannot be detected in the table. In this case, we miss important information that is necessary for the correct normalization of the table.

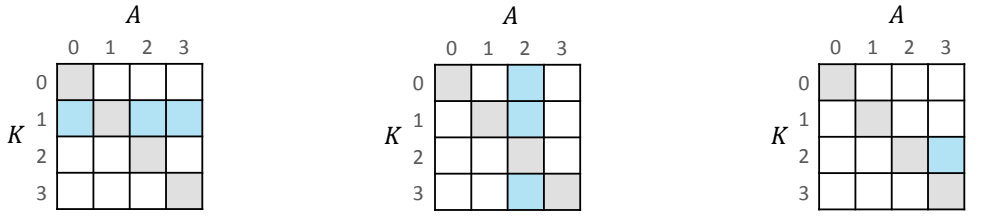
2. **Coincidental Functional Dependencies:** Additional, spurious functional dependencies that do not represent meaningful relations, are inferred from the data. Caused by coincidental data correlations, these dependencies also lead to incorrect normalization results.

To address these challenges adequately in our normalization approach, it is important to identify the reasons behind these scenarios. Recalling the characteristics of tables on the Web that we specified in

City	Mayor	Time Zone	Country
	B. Johnson		
	O. Quinn		
	K. Wowereit		
	B. Delanoë		

City	Mayor	Time Zone	Country
		CET	
		CET	
		CET	
		CET	

City	Mayor	Time Zone	Country
		GMT	England
		EST	USA
		GMT	England
		PST	USA



(a) Unique Attribute (b) Constant Attribute (c) Spurious Correlation

Figure 6.2: Causes for coincidental functional dependencies $K \rightarrow A$.

Section 3.1, we can establish four main characteristics that contribute negatively to the inference of functional dependencies. In the following paragraphs, we take a closer look at each of these characteristics.

Non-descriptive Attribute Labels
To infer meaningful relations between attributes from the table header, we require labels that provide a clear description of the attribute semantics. However, the quality of attribute labels in Web tables varies greatly and the semantics is often not clear enough to directly infer relations. As we highlighted previously, we frequently encounter missing labels. If labels are provided, they are often difficult to interpret, if they contain abbreviations or acronyms, generic terms or lengthy descriptions.

Insufficient Sample Size
Tables on the Web, especially those embedded in HTML, frequently contain only a small number of tuples. For the inference of correlations between attributes, the tuples become the samples the inference is based on. In general, this means that very small tables do not provide enough samples to derive any statistically significant correlations. For functional dependencies in particular, we see the effect that the smaller the sample size, the more likely it is to encounter coincidental functional dependencies. Depending on the distribution of attribute values in the table, we distinguish three scenarios, which are pictured in Figure 6.2. For each scenario, a table with a sample of possible values illustrates the characteristics of the value distribution and a dependency matrix shows the resulting meaningful and coincidental dependencies.
A *unique attribute* is an attribute in the table that is not the key of the relation, but takes on a *distinct value* for each tuple, similar to a key. Such a characteristic is very common for numeric attributes, but the smaller the sample size, the more likely it is for non-numeric attributes, such as character strings or dates to show only distinct values. This is a challenge for the detection of the correct key of a relation. Considering functional dependencies, a unique attribute always *determines* every other attribute in the table, as illustrated in the dependency matrix in Figure 6.2(a). Without redundancy in the data, it is difficult to detect relevant dependencies. Generally, the more redundancy we have in the data,

Table 6.1: Example tables showing inconsistencies in the body.

(a)

Program Name	Program Address	Phone Number
Beth Emeth Homecare	1080 **McDonald** Avenue	**(718)** 253-2220
Beth Emeth Homecare	1080 **MacDonald** Avenue	**718** 253-2220

(b)

Name	Address	Borough
New Fulton Fish Market Cooperative	800 Food Center Drive **Unit** 65B	Bronx
New Fulton Fish Market Cooperative	800 Food Center Drive **Room** 65B	Bronx

the more evidence we have to support meaningful dependencies and refute spurious ones (Ilyas et al., 2004).
A *constant attribute* represents the opposite of a unique attribute. Here, the attribute takes on the *same value* for every tuple in the table. In many cases, such an attribute would be factored out to the context of the table. However, if a constant attribute is kept in the table, it becomes a challenge for the inference of functional dependencies. As illustrated in the dependency matrix, a constant attribute is always *dependent* on any other attribute in the table. Without any variation in the values, there is no evidence to detect spurious correlations.
Finally, coincidental dependencies can occur despite redundancy and variation in the data. As illustrated in Figure 6.2(c), *spurious correlations* can arise between any pair of attributes that are neither unique nor constant. In contrast to the first two scenarios, such correlations only affect the two attributes involved, instead of the entire attribute set.

Inconsistent Tuples
Compared to relational data managed in database management systems, relational data on the Web is more often affected by errors or inconsistencies. With no centralized data design and quality control mechanisms in place, there is no verification process to ensure data integrity (Cafarella et al., 2011b). As a result, there is a good chance that we encounter inconsistencies in the data that violate the functional dependency constraints. In Table 6.1, we show three common types of such inconsistencies.

- **Orthographic Variations:** The second column in Table 6.1(a) shows an example for an inconsistency caused by a difference in the spelling of a street name. From the context it is clear that in this case both names refer to the same street. However, in other cases, different spelling conventions can also mean that the data refers to separate entities.

- **Presentational Variations:** The third column in Table 6.1(a) shows an irregularity that results from the same value being presented in different formats. While the similarity is obvious, when presented to a human user, internally both entries are represented by different values. Algorithmically, it is very challenging to distinguish between characters that are essential to the meaning of the value and characters that are merely used for presentational purposes.

- **Linguistic Variations:** An example for an inconsistency caused by the use of synonyms is depicted in Table 6.1(b). In the second column, the words *unit* and *room* are used interchangeably to refer to the same address.

Such inconsistencies, which are most likely the result of deficient data management practices, are very difficult to identify and resolve algorithmically. They cause meaningful functional dependencies to become undetectable in the table, if they occur in the dependent attribute and, thus, violate the dependency constraint.

Anonymization of Personal Data
When personal or enterprise data is published on the Web, sensitive information, such as names or contact details, is sometimes anonymized in order to protect the interests of the individual or company in question. For instance, internal government data published on Open Data portals is frequently edited to prevent certain information from becoming public knowledge. In the tables, the respective values are missing or replaced by default placeholders. Since public datasets on the Web are authored and published by many different people, there is no global convention regarding anonymization. Many different placeholders are used in Web tables. Even within the same table, placeholders can vary. Common examples for such placeholders include *N/A*, *undisclosed* or *redacted*. By removing the original values or replacing them with default values, the distribution and correlation of values within the table are modified significantly. As a result, anonymization can cause meaningful functional dependencies to be undetectable and coincidental dependencies to be inferred.

6.4 NORMALIZATION PROCESS

As outlined previously, to (semantically) normalize multi-concept Web tables, we require meaningful functional dependencies that reflect the semantics of the table data. Relying on expert support and domain knowledge to derive functional dependencies is too extensive and unrealistic considering the scale and domain coverage of a Web table corpus. Instead, we need to infer these dependencies directly from the data. However, as illustrated in the previous section, due to the limited sample size of individual Web tables, inferring functional dependencies from the data also introduces incorrect dependencies. Consequently, we need to filter the resulting candidate set in order to retrieve only *meaningful* functional dependencies.
We propose a systematic approach, tailored to the characteristics of Web tables, to *extract*, *evaluate* and *filter* functional dependencies. Figure 6.3 provides an overview of the process involved. In the first processing step, we extract all necessary information from the table in order to form an *attribute dependency graph*. This graph is composed of all functional dependencies that hold between pairs of attributes in the table. If we consider all possible attribute combinations when inferring functional dependencies, we include too many spurious correlations. As a result, identifying relevant functional dependencies becomes very challenging. Instead, we first identify the number of concepts presented in the table via entity columns. Then, based on these entity columns, we extract inter-concept and intra-concept dependencies, which correspond to EEB and EAB relations, respectively. This approach significantly reduces the number of candidate dependencies. After establishing the dependency graph, we systematically evaluate and filter the dependencies (processing step 2), in order to reduce the graph to only contain meaningful functional dependencies that correctly characterize

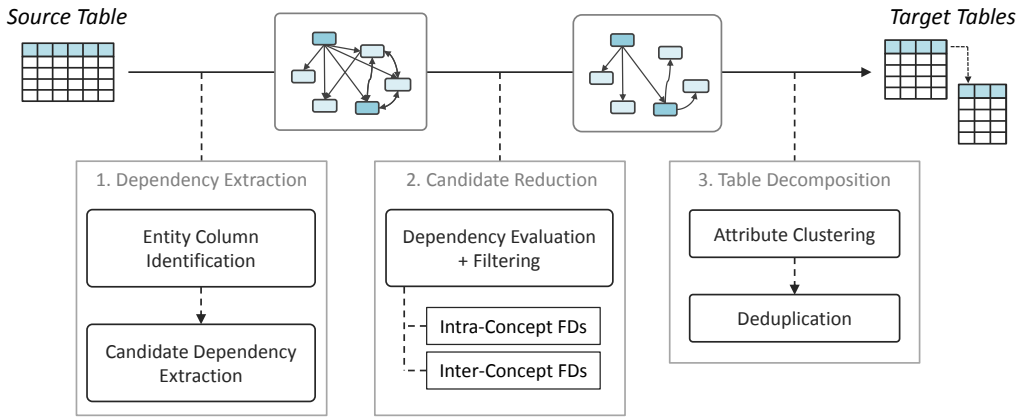


Figure 6.3: Process overview: Extraction, evaluation and filtering of functional dependencies inferred from table data.

Algorithm 6.1 Normalization of relation R of a multi-concept table.

function NORMALIZE(R)
  $relations \leftarrow \{\}$
  C_{entity} =GETENTITYCOLUMNS(R)
  $intraFD, interFD$ =GETFUNCTIONALDEPENDENCIES(R, C_{entity})
  $intraFD_{final}$ =EVALUATEANDFILTER($intraFD$)
  $interFD_{final}$ =EVALUATEANDFILTER($interFD$)
  $relations$ =DECOMPOSE(R, $intraFD_{final}$,$interFD_{final}$)
  return $relations$
end function

the semantic concepts in the table. These remaining dependencies are then utilized in the final step to identify concept boundaries and split the source table accordingly. In the subsequent sections, we provide a detailed description of each processing step, including the challenges involved.

6.4.1 Entity Column Identification

In Web tables, concepts typically feature an attribute that stores the names of the instances of the concept and serves as a *natural key* for these instances. In the context of Open Data tables, which are often exported from active data management systems, concepts sometimes alternatively feature *surrogate keys*, artificial identifiers to ensure uniqueness, if it is not guaranteed by the natural key, or if no natural key exists. We denote these key-like attributes, which name individual entities in the tables, as so-called *entity columns*. From these names or identifiers, as well as other related attributes, we can identify the individual concepts mentioned in the table. Therefore, identifying entity columns in the table is an important step towards recognizing concepts.
In tables that only describe a single concept, the entity column provides a good indication of the table's main topic. As a result, in the context of single-concept tables, entity columns are also referred

to as a table's *subject column*. In the literature, several techniques have been proposed to detect entity columns in single-concept tables. Cafarella et al. simply regard the left-most column of a Web table as a *semantic key* that summarizes each row (Cafarella et al., 2008a). This very simple heuristic is extended by Venetis et al. to regard the left-most column that contains neither numbers nor dates (essentially, a string column). This simple rule already achieves an accuracy of 83% in an experimental evaluation (Venetis et al., 2011). The same authors also propose a classification-based approach that employs support vector machines to further improve the accuracy to 94%.
An alternative approach is proposed by J. Wang et al. (2012). Entity columns are detected by matching the table to the external knowledge base *Probase*, which models concepts associated with common attributes as well as instances. Two conditions are defined to identify entity columns: (1) the column should contain entities of the same concept, and (2) the remaining attributes should best describe the same concept. The column that best meets both conditions is selected as the table's entity column. In contrast to a learning-based approach that relies on intrinsic table characteristics, the effectiveness of this approach by J. Wang et al. is closely tied to the coverage of the external source.
So far, entity column identification has focused solely on single-concept tables. In this thesis, we apply a learning-based approach that is similar to the classification approach proposed by Venetis et al. (2011), but is directed at multi-concept tables. Venetis et al. use a binary classifier and a set of five features to automatically detect entity columns. In single-concept tables, the entity column most frequently coincides with the left-most column in the table that contains string values. The proposed features reflect this correlation. In detail, Venetis et al. use the following features:

1. **Unique Values:** In single-concept tables, the entity column describes the *subject* of the table and, as a result, frequently serves as the *key* attribute. As the key for the table, all values of the entity column are required to be distinct.

2. **Numeric Values:** Entity columns in public tables more often contain natural keys, i.e. names or titles, instead of surrogate keys. Consequently, numeric content indicates that a column is *not* a potential entity column.

3. **Variance:** Natural keys, such as names, are generally very homogeneous regarding the number of words or tokens per entry. For instance, names of people mostly consist of first name and last name and occasionally one or more middle names. Overall, almost all entries follow a similar pattern. In contrast, other string attributes, most notably *notes* or *comments*, often feature a much greater variance.

4. **Average Content Size:** Similar to the variance, the absolute number of tokens per entry also distinguishes entity names from other string attributes. Compared to potentially long entries in *comments* or *address* columns, entity names are generally much shorter.

5. **Column Index:** As mentioned before, the entity column in a single-concept table is frequently found in the left-most string column. As a result, lower index numbers are a good indicator for entity columns.

Some of these features, specifically the uniqueness of values and the column index, are specifically tailored to the characteristics of single-concept tables and, as such, are not sufficient to identify entity columns in more diverse multi-concept tables. In these tables, the entity columns can be positioned

Table 6.2: Classification features, including features 1-5 proposed by Venetis et al. (2011).

No.	Feature Description	No.	Feature Description
1	Fraction of cells with unique content	6	Fraction of cells with string content
2	Fraction of cells with numeric content	7	Relative column position
3	Variance in number of data tokens	8	Has column label
4	Average number of data tokens	9	Label contains *name* or *id*
5	Column index (from the left)	10	Correlation with left neighbor
		11	Correlation with right neighbor

anywhere in the table (although it is rare to find entities in the very last column). Additionally, not all entity columns necessarily have distinct values. Unless a one-to-one relationship exists between all pairs of concepts, some entity columns will contain duplicate entries due to one-to-many relationships. To address these differences in characteristics, we extend the original set with six additional features:

6. **String Values:** While numeric values are already addressed in the second feature, we check for string values to make a more detailed distinction between string columns and other non-numeric data types, such as dates.

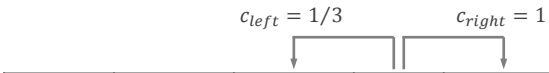
7. **Relative Column Position:** The number of attributes in public tables varies greatly. As a result, the same column index could point to the last column in one table and to a column in the first half of another table. Therefore, we consider column positions relative to the overall table size.

8. **Label:** Entity columns represent key attributes for the individual concepts and, as such, carry important information. Therefore, we expect these columns to be more likely to be given an attribute label than non-key attributes.

9. **Key Indicators:** In many cases, special keywords or phrases in the attribute labels indicate entity columns. The most prominent of these terms are *name* and *id*. Such indicators are especially useful for attributes that otherwise do not satisfy the general assumptions for entity columns, such as numeric identifiers.

10. **Correlation with Left Neighbor:** Assuming a general reading order from left to right, with attributes of a concept listed *after* its corresponding entity column, we can utilize attribute correlation to identify potential entity columns, as illustrated in Figure 6.4. The correlation score, specified in Equation 6.1, determines the strength with which the values in an entity column correlate with the values in a neighboring column. It is set to zero if no such neighbor exists, i.e. at the beginning and end of the table. If the attribute directly to the left of the entity column belongs to a different concept, we expect only a weak correlation.

11. **Correlation with Right Neighbor:** Similar to the correlation with the left neighbor, we consider the correlation with the right neighbor. In many cases, the attribute directly to the right of an entity column belongs to the same concept. As a result, there is a functional dependency between the columns and, therefore, a strong correlation between the values.



City	Mayor	Elevation(m)	Country	Area (km²)	Population (Mio.)
London	B. Johnson	35	England	130,395	53
Manchester	N. u. Hassan	38	England	130,395	53
Liverpool	J. Anderson	70	England	130,395	53
Dublin	O. Quinn	6	Ireland	70,273	4.59
Cork	M. Shields	15	Ireland	70,273	4.59
Galway	D. Lyons	9	Ireland	70,273	4.59

Figure 6.4: Correlation of an entity column with neighboring columns to the left and right.

Based on these features, we train a classifier using state-of-the-art techniques, to identify potential entity columns. Binary classification techniques that can be applied include support vector machines, decision trees or random forests. A detailed evaluation of the effectiveness of the incorporated features is presented in Section 6.5.1.

Clustering Related Entity Columns

So far, we assumed a single entity column for each concept in a table. However, in real-world tables, we occasionally encounter multiple columns that are suitable entity column candidates for the same concept, as shown in Figure 6.5. In most cases, the concept in question has an attribute containing natural keys (i.e. names) and an attribute containing generated surrogate keys (i.e. artificial identifiers). The additional ids are generally used to ensure a unique reference for each entity in cases where the name is not necessarily unique. In our normalization approach, we expect each entity column to represent a different concept. To avoid splitting a concept with two potential entity columns into two separate concepts, we need to identify these *related* entity columns, combine them and select one representative as the entity column for the concept that is used throughout the normalization process.
We identify related entity columns utilizing a simple heuristic. Considering only the attribute labels of all candidate entity columns, we parse the labels and construct a prefix tree from the labels split at word boundaries. Pairs of labels with the same label prefix, which often coincides with the name of the concept, and suffixes *name* and *id* are clustered to refer to the same concept. As the representative for the concept, we select the attribute with the highest cardinality. If both have the same cardinality, we select the left-most candidate. For the remainder of the normalization process, only the representative column is used. The remaining candidate is simply regarded as one of the attributes describing the concept.

Selecting Main Entity Column

In multi-concept tables, we frequently encounter one concept that describes the main topic of the table, i.e. what the table is about. The other concepts are merely supplementary to provide a context for the topic. For instance, the example table depicted in Figure 6.5 is mainly about cities, while the concepts mayor and country simply complement the topic. We refer to the entity column representing this concept as the *main* entity column of the table. In many cases, this entity column serves as a key

City ID	City Name	Mayor	Elevation (m)	Country	Population (Mio.)
C1	London	B. Johnson	35	England	53
C2	Manchester	N. u. Hassan	38	England	53
C3	Dublin	O. Quinn	6	Ireland	4.59
C4	Cork	M. Shields	15	Ireland	4.59
C5	Galway	D. Lyons	9	Ireland	4.59

Figure 6.5: Example table containing a concept *City* with two potential entity columns, a natural key *City Name* and a surrogate key *City ID*.

for the table, given a simple key exists. Based on this correlation and a general reading order of the table from left to right, we use the following rule to identify the main entity column. From the set of entity columns identified in the table, we select the left-most candidate with the largest number of functionally dependent columns.
For tables that express a relationship between equally relevant concepts and, as a result, do not have a designated *main* concept, we still select one. However, doing so has no significant impact on the normalization of these tables.

6.4.2 Candidate Dependency Extraction

After identifying all entity columns that represent a unique concept in the table, we extract functional dependencies to assemble an initial *attribute dependency graph*. Recognizing all relevant dependencies in the data, i.e. achieving a high recall, is essential, as all of these dependencies are required to normalize the source table correctly. Missing correct dependencies will ultimately lead to incorrect concept boundaries.

Probabilistic Functional Dependencies
In general, we expect *exact* functional dependencies $X \rightarrow A$, meaning that the dependency holds for each non-NULL value $x \in X$. However, as illustrated in Section 6.3, inconsistencies in the data, which are frequent in public tables, can lead to a violation of this constraint. As a result, we miss some important dependencies.
To address these inconsistencies in the data, instead of calling for exact dependencies, we need to consider approximate functional dependencies. In the literature, several measures to specify some notion of *exactness* for functional dependencies have been proposed. Huhtala et al., for example, consider the minimal number of entries in the table that need to be removed for the functional dependency to hold (Huhtala et al., 1999). The more entries violate the constraint, the less exact is the functional dependency. Alternatively, Ilyas et al. use the ratio $|X|/|X, A|$, where $|\cdot|$ denotes the number of distinct values, to measure the exactness of what they refer to as *soft* functional dependencies (Ilyas et al., 2004). This measure returns its highest possible value of one for exact dependencies, with lower values indicating violations of the dependency constraint.

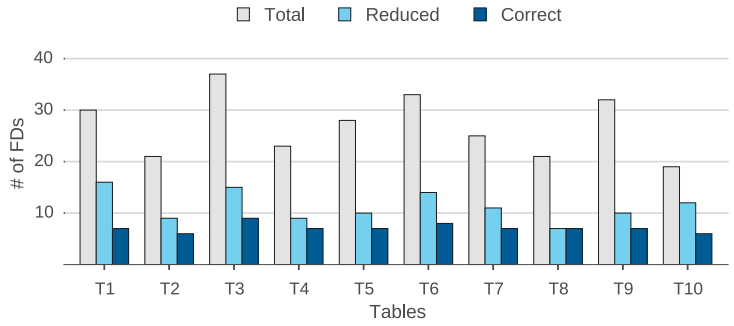


Figure 6.6: Comparison between all functional dependencies that can be inferred from a table (*total*) and the amount of dependencies after systematic extraction (*reduced*). The number of dependencies manually labeled as valid dependencies are marked in dark blue (*correct*).

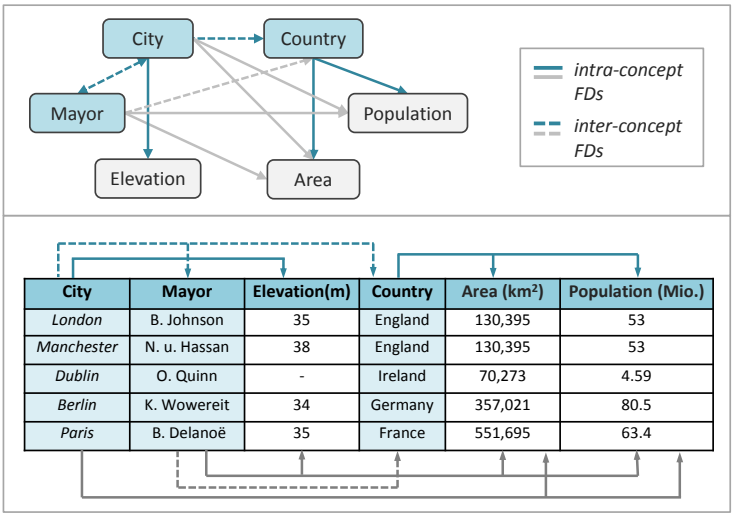
In this thesis, we employ the notion of *probabilistic* functional dependencies introduced by D. Z. Wang et al. (2009). A probabilistic FD is denoted as $X \xrightarrow{p} A$, where p is the probability that $X \rightarrow A$ holds in the data. The probability is calculated as follows, where D_x is the set of distinct values in X and a_x is the value in A that occurs most often in conjunction with value x (considered the *true value* of A):

$$Pr(X \rightarrow A) = \frac{\sum_{x \in D_x} |x, a_x|}{\sum_{x \in D_x} |x|} \tag{6.1}$$

The overall probability is derived from the weighted average of the probabilities for each distinct value $x \in D_x$. The weights are determined by the frequency at which x occurs in X. Consequently, the probability of $X \rightarrow A$ holding for rare values of X has less impact on the overall score than the probability of $X \rightarrow A$ holding for frequent values.
Similar to other approaches based on approximate FDs, a threshold θ_p determines how much noise or inconsistency is tolerated in the data. The optimal threshold strongly depends on the characteristics (especially the quality) of the data. When selecting a threshold value, we are faced with a trade-off between tolerating constraint violations in order to detect relevant dependencies and ignoring such violations and, as a result, introducing incorrect dependencies.

Functional Dependency Inference
Given the assumption of simple keys and the definition of probabilistic functional dependencies, we can now infer probabilistic functional dependencies $X \xrightarrow{p} A$, with $p \geq \theta_p$, from the data. Instead of following the naive approach of considering all functional dependencies that can be inferred from the data, we only consider the following dependencies:

- **Inter-Concept Functional Dependencies:** First, we consider functional dependencies between concepts, represented by their respective entity columns. For each pair of entity columns, we evaluate the dependency constraint and extract FDs accordingly. Inter-concept FDs can be bi-directional, reflecting a $1:1$ relationship between the concepts.



City	Mayor	Elevation(m)	Country	Area (km²)	Population (Mio.)
London	B. Johnson	35	England	130,395	53
Manchester	N. u. Hassan	38	England	130,395	53
Dublin	O. Quinn	-	Ireland	70,273	4.59
Berlin	K. Wowereit	34	Germany	357,021	80.5
Paris	B. Delanoë	35	France	551,695	63.4

Figure 6.7: Example illustrating the classification of functional dependencies inferred from the data.

- **Intra-Concept Functional Dependencies:** Second, we consider functional dependencies between attributes of the same concept, specifically dependencies between the concept's entity column and other non-key attributes. Therefore, we evaluate the dependency constraint for each pair of attributes consisting of an entity column and one of the remaining attribute columns. We only consider functional dependencies where an entity column is the determinant attribute.

The dependency graph resulting from the inference algorithm reflects an initial specification of the functional dependencies holding in the table. Compared to the naive extraction approach, the number of potential functional dependencies is significantly reduced following our systematic approach, as we do not consider unlikely candidates, such as dependencies between non-key attributes. Figure 6.6 quantifies this reduction for ten example tables from our test set. For most tables, we can reduce the set of candidate functional dependencies by more than 50%.
Figure 6.7 illustrates the derived attribute dependency graph for an example table. As we can see, the initial graph still contains *transitive* dependencies. Transitivity is one of three properties or inference rules for functional dependencies known as *Armstrong's axioms* (Armstrong, 1974). It states that, if functional dependencies $X \rightarrow Y$ and $Y \rightarrow Z$ hold in the data, then $X \rightarrow Z$ holds as well. In multi-concept tables, transitive FDs occur very frequently, as a direct result of the join of multiple concepts. Furthermore, the dependency graph can also contain *coincidental* or *spurious* functional dependencies. As highlighted in Figure 6.2, spurious FDs occur due to a table representing only a limited subset of an attribute domain. The potentially small sample size that is characteristic for Web tables means that there is not enough evidence in the data to discard invalid dependency. The more samples are available, the less likely it is for spurious FDs to occur, as the table becomes a better representation of the domain. To identify all valid functional dependencies in the graph, we need to remove both transitive as well as spurious dependencies.

6.4.3 Indicators for Attribute Relatedness

In order to distinguish between meaningful and spurious functional dependencies inferred from the data, we rely on a set of structural and semantic measures that indicate valid relationships between attributes. The structural measures *Column Distance*, *Value Correlation* and *Null Value Distribution* address the column arrangement as well as the distribution of values in the table. The semantic measures *Common Concept Label* and *Attribute Co-Occurrence* take into account the attribute labels and their semantic relationships.

Column Position and Distance
For each functional dependency $X \rightarrow A$, we consider the position of dependent attribute A relative to the determinant X with respect to *order* and *distance*. This is motivated by the following two assumptions. First, we assume that, following the general reading order from left to right, the concept identifier is more likely to be mentioned first, *before* any related attribute is listed. As a result, functional dependencies where determinant X appears before dependent A in the table should receive a higher score than dependencies with X appearing after A. Second, we assume that attributes that describe the same concept appear in close proximity in the table to indicate their relatedness. Thus, a higher score is assigned to FDs where the absolute distance between X and A is small.
We combine both assumptions into a single score, as specified in Equation 6.2.

$$score_{CD}(A, X) = \begin{cases} 1 - \dfrac{dist(X, A)}{\theta_d} & \text{, if } dist(X, A) < \theta_d \\ 0 & \text{, else} \end{cases} \tag{6.2}$$

$$dist(A_i, A_j) = \begin{cases} j - i & \text{, if } i < j \\ i - 1 & \text{, else} \end{cases} \tag{6.3}$$

An adjustable threshold θ_d is used to determine the overall score. We consider two scenarios to determine the threshold. First, we consider a global threshold, that applies for all tables, for example the average or maximum number of columns in the tables. Second, we consider a local threshold, such as the number of columns in the respective table, to retrieve a score that is relative to the table size. Note that, in order to compare scores across tables, a global score is preferable.

Value Correlation
In addition to functional dependencies, we can take other correlation measures into consideration to evaluate the relatedness of attributes. Specifically, we use *variation of information* (VI) to measure *redundancy* between attributes X and A. Multi-concept tables frequently contain redundancy in the data as a result of a $n : 1$ relationship between concepts (see example in Figure 6.1). A strong correlation in the redundancy of two attributes indicates a strong relatedness, as there is more *evidence* that the functional dependency is not spurious. Ilyas et al. refer to this notion as the amount of *information* that is present in the table. They regard a functional dependency $X \rightarrow A$ as more likely to be correct, if the cardinality of a table T is significantly larger than the joint cardinality of columns A and X, denoted as $|T| \gg |A, X|$ (Ilyas et al., 2004).

Equation 6.4 specifies variation of information as a distance metric, where $I(A, X)$ is the *mutual information* between columns X and A. $H(A)$ and $H(A|X)$ denote the *entropy* and *conditional entropy*, respectively.

$$VI(A, X) = H(A) + H(X) - 2I(A, X) \tag{6.4}$$

- **Mutual Information:**
$$I(A, X) = H(A) - H(A|X)$$
- **Entropy:**
$$H(A) = -\sum_{a \in \mathcal{A}} p(a) \log p(a)$$
- **Joint Entropy:**
$$H(A, X) = -\sum_{a \in \mathcal{A}, x \in \mathcal{X}} p(a, x) \log p(a, x)$$
- **Conditional Entropy:**
$$H(A|X) = -\sum_{a \in \mathcal{A}, x \in \mathcal{X}} p(a, x) \log \frac{p(a, x)}{p(x)}$$

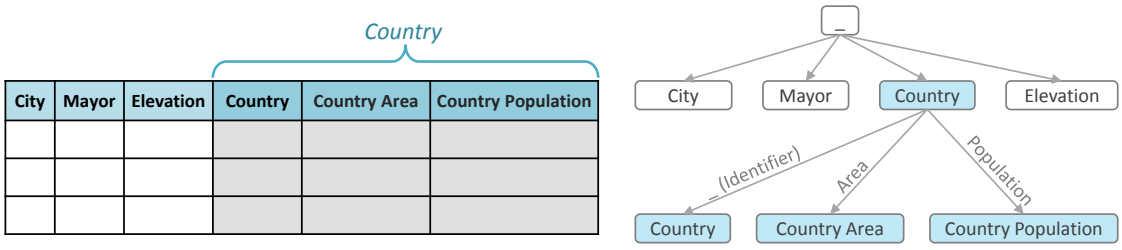
From the variation of information, we derive a normalized similarity score in the range $[0, 1]$ following Equation 6.5. Note that in the special case where both columns A and X are *constant*, with $H(A) = H(X) = 0$, this score is undefined. While this is reasonable from an information theoretic point of view, where a source S with entropy $H(S) = 0$ is considered to provide *no* information, it requires special consideration in the context of tables. The data generally presents only a subset of the attribute domain, as an attribute domain that contains only a single value is very rare. Consequently, two constant attributes are very likely related and should be interpreted as any other case where $H(A) = H(X)$. As a result, in this case, we assign a score of one.

$$score_{VI}(A, X) = 1 - \frac{VI(A, X)}{H(A, X)} \tag{6.5}$$

Distribution of NULL Values

Tables on the Web frequently contain missing information, which is treated as *NULL* values when processing the table. In the closed world of a relational database, NULL indicates that a certain value does not exist. In public data, however, there are many different reasons for missing values. In addition to property values that do not exist, in some cases the information was simply not known to the author of the data or removed deliberately in order to conceal information.

Functional dependencies $X \rightarrow A$ are only defined for entries $x, a \neq NULL$, with $x \in X, a \in A$. That means that, so far, NULL values and their distribution in the table have not been considered for the inference of functional dependencies. However, the distribution of NULL values can help us identify coincidental dependencies. For a meaningful dependency $X \rightarrow A$, we expect key column X



(a) Example Schema (b) Prefix Tree

Figure 6.8: Example schema containing a concept *Country* with the respective concept label. The prefix tree highlights the attributes that share this common concept label.

to *not* be NULL for all instances where attribute column A is *not* NULL. Otherwise, X may not be the correct key for the attribute.
Based on this assumption, we assign a score in the range $[0, 1]$ as specified in Equation 6.6. N denotes the number of rows in the table, while $x_i \in X$ and $a_i \in A$ denote the ith entries in the respective columns.

$$score_{NV}(A, X) = \frac{\sum_{n \in N} g(x_n)}{\sum_{n \in N} g(x_n) \cdot g(a_n)} \tag{6.6}$$

$$g(x) = \begin{cases} 1 & \text{, if } x \neq NULL \\ 0 & \text{, else} \end{cases} \tag{6.7}$$

Common Concept Labels
In addition to indications for attribute relatedness derived from the table data, we can also find hints in the description of the table, i.e. the attribute labels. Analyzing the characteristics of tables on the Web, we noticed that in many cases attributes of the same concept share a common label to indicate their relatedness. Figure 6.8 shows an example table header with a concept label *Country*.
Apart from different formatting conventions, in the vast majority of cases, concept labels form a prefix to the actual attribute label. To identify such *common concept labels* (CCL), we create a prefix tree from the attribute labels, splitting at word boundaries. This approach is similar to the prefix tree used to cluster entity columns, as described in Section 6.4.1. As common concept labels, we only consider prefixes that are shared by at least two attributes. Furthermore, for each pair of attributes, we are only interested in the *longest common prefix*.
To ensure valid concept labels, we enforce the following two constraints: (1) the concept label, i.e. the common prefix, must be a *noun* or *compound of nouns*, and (2) the remaining attribute label must be a valid *noun phrase*. Cardinal numbers and single letters are not permitted, as they do not reflect meaningful attribute labels. A special case is formed by labels that only contain the concept label, without an additional attribute label. This is frequently the case for entity columns, which contain the *name* or *identifier* of instances of the concept. Instead of using, for example, the label *Country Name*, the column is simply labeled as *Country*. In this case, we append the temporary suffix *identifier* to the attribute label, before computing the similarity score.

Algorithm 6.2 Function to compute coherency score for a schema T, as described by Cafarella et al. (2008a).

```
1: function COHERENCY(T)
2:     totalPMI ← 0
3:     for all a ∈ T, b ∈ T, a ≠ b do
4:         totalPMI ← totalPMI + PMI(a, b)
5:     end for
6:     return totalPMI/(|T| ∗ (|T| − 1))
7: end function
```

In order to identify nouns and noun phrases in the attribute labels, we use a lexical parser to verify the word category (part of speech) of each word. A score is only computed for column pairs sharing a common prefix, where both conditions hold. All remaining pairs receive a score of zero.
In some instances, where tables contain multiple nested concepts, the respective concept labels can form a hierarchy. Consider, for example the following schema *(Registration Office, Date, Vehicle Reg. Nr., Vehicle Owner, Vehicle Owner Address)* for a table that describes car registration data. The concept labels *Vehicle* and *Vehicle Owner* form a hierarchy. In a case like this, a longer common prefix suggests a stronger semantic relatedness, and, as a result, should receive a higher score. In our example, the attributes *Vehicle Owner* and *Vehicle Owner Address* are more closely related than *Vehicle Reg. Nr.* and *Vehicle Owner Address*. To incorporate this notion, we compute the common concept label score following Equation 6.8, where $CCL(A, X)$ denotes the common prefix shared by attributes X and A and $maxCCL(A)$ denotes the longest prefix attribute A shares with any attribute in the table. To measure the length of a prefix, we use the word count.

$$score_{CCL}(A, X) = \frac{|CCL(A, X)|}{|maxCCL(A)|} \tag{6.8}$$

Attribute Co-Occurrence
Concept labels are generally not available for all attributes that we consider. Therefore, we incorporate an additional, more general measure to evaluate the semantic relatedness between attributes. It is based on the *coherency score* for Web table schemas proposed by Cafarella et al. (2008a). The *coherence* of a schema is defined as the average relatedness of pairs of attributes in the schema, as shown in Algorithm 6.2. If all attributes are strongly related to all other attributes in the schema, the schema is considered *coherent*. Intuitively, single-concept tables, where all attributes describe the same concept, are very coherent, whereas multi-concept tables are less coherent.
The relatedness between a pair of attributes is calculated using *point-wise mutual information* (PMI), defined in Equation 6.9. This measure returns a positive value for correlated variables, zero for independent variables and a negative value for negatively correlated variables. The probability scores required to compute the PMI are derived from a large collection of schema statistics. Cafarella et al. collected attribute and schema frequencies from a large corpus of Web tables, published as the *Attribute Correlation Statistics Database* (ACSDB). We also utilize this database to derive attribute probabilities and compute PMI scores for all candidate column pairs (A, X).

$$PMI(A_1, A_2) = \log \frac{p(A_1, A_2)}{p(A_1)p(A_2)} \tag{6.9}$$

$$nPMI(A_1, A_2) = \frac{PMI(A_1, A_2)}{-\log p(A_1, A_2)} \tag{6.10}$$

In order to derive probability scores from ACSDB, we must match the attributes we encounter in a table to the attributes contained in the collection. To increase the chance of finding corresponding attributes and to address the ambiguity and sometimes low quality of attribute labels in Web tables, we apply basic schema normalization techniques (Sorrentino et al., 2009), as well as synonym expansion. To receive a similarity score in the range $[0, 1]$, we first use a normalized variant of point-wise mutual information, denoted as $nPMI$, which scales the PMI score to the range $[-1, 1]$. Furthermore, we are only interested in positive correlations between the attributes. Therefore, we specify the similarity score as depicted in Equation 6.11. If the normalized PMI of attributes A and X is positive, we return it as the similarity score. Otherwise we return a score of zero, indicating that no positive evidence suggesting a semantic relatedness between the attributes was found.

$$score_{AC}(A, X) = \begin{cases} nPMI(A, X) & \text{, if } nPMI(A, X) > 0 \\ 0 & \text{, else} \end{cases} \tag{6.11}$$

6.4.4 Dependency Filtering and Decomposition

From the table, we inferred an attribute dependency graph that contains meaningful, but also transitive as well as spurious dependencies. Using the indicators for attribute relatedness described in the previous section, we can now evaluate these dependencies in order to filter out incorrect or unnecessary ones. The remaining dependencies then form the input for the decomposition algorithm.

Combining Similarity Scores

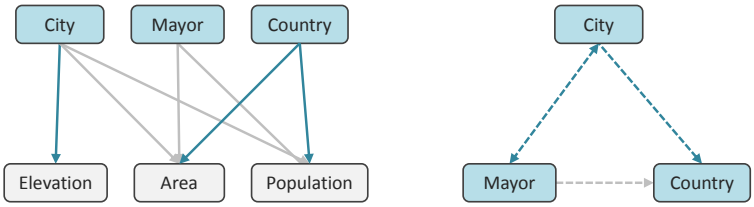
To score each functional dependency $X \rightarrow A$ in the dependency graph, we combine all individual similarity scores using a weighted sum as specified in Equation 6.12.

$$\begin{aligned} score_{total}(A, X) = \omega_0 &+ \omega_1 \cdot score_{AC}(A, X) + \omega_2 \cdot score_{CCL}(A, X) \\ &+ \omega_3 \cdot score_{NV}(A, X) + \omega_4 \cdot score_{VI}(A, X) \\ &+ \omega_5 \cdot score_{CD}(A, X) \end{aligned} \tag{6.12}$$

Inter- and intra-concept dependencies feature very different characteristics. As a result, not all indicators are equally useful for both types. For instance, common concept labels are much more prominent for intra-concept dependencies. We address these differences by using a separate set of weights for each type. The specific weights are inferred from pre-labeled training data using linear regression.

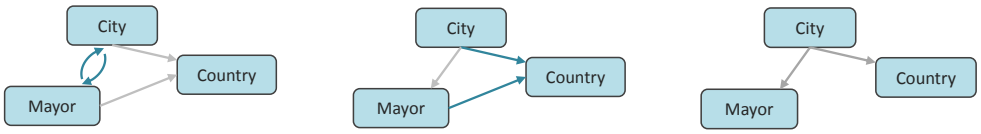
Intra-Concept Dependencies

First, we consider only intra-concept functional dependencies, which reflect a semantic connection between an entity column and an additional (non-key) attribute. For each attribute A in the table that has not been identified as an entity column, we retrieve all entity columns X with $X \rightarrow A$.



(a) Intra-Concept FD (b) Inter-Concept FD

Figure 6.9: Example attribute dependency graph depicted in Figure 6.7, separated into intra- and inter-concept dependencies. Entity columns are marked in blue, all remaining columns in grey.

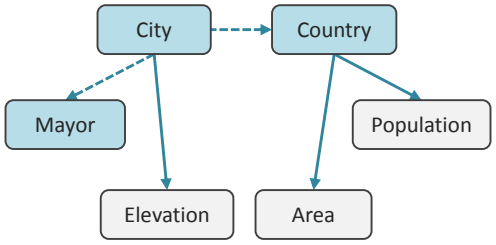


(a) (b) (c)

Figure 6.10: Selection process for inter-concept dependencies.

Our goal is to find the entity column that represents the concept attribute A *most likely* belongs to. As illustrated in Figure 6.9(a), for some attributes, we retrieve several potential entity columns. We score and rank these candidates, in order to select the highest-ranking candidate as the most likely entity column.
In the case where more than one candidate receives the highest score, we break the tie using the following procedure. If one of the candidates is the *main* entity column, we select this candidate as the best candidate. If not, we resort to the *Column Distance* score to decide. As a result, we always select exactly one entity column as the concept key for each non-key attribute.

Inter-Concept Dependencies
In the next step, we consider only inter-concept functional dependencies, which reflect relationships between two concepts, represented by dependencies between the entity columns of each concept. The relationships between concepts can be much more complex than the dependencies within a concept. In general, we distinguish between $1:1$, $n:1$ and $n:m$ relationships. In this approach, we focus on $1:1$ and $n:1$ relationships. $n:m$ relationships are not that easily detected in the data, as they are not reflected through functional dependencies. As a result, detecting $n:m$ relationships in multi-concept tables requires a more comprehensive semantic analysis, which is subject to future work.
As illustrated in Figure 6.9(b), we collect all functional dependencies between entity columns. After scoring each dependency, we follow the steps depicted in Figure 6.10 in order to filter out incorrect dependencies. First, we resolve bi-directional dependencies between concepts. Although they may represent a valid $1:1$ relationship between the concepts, we need to remove one direction in order to avoid splitting the dependency graph into unconnected subgraphs. The directionality of the relation can be restored later. If one of the entity columns involved in a bi-directional dependency is the main entity column of the table, we always favor the functional dependency with the main entity column



(a) Dependency Graph

City	Mayor	Elevation(m)	Country
London	B. Johnson	35	England
Manchester	N. u. Hassan	38	England
Dublin	O. Quinn	-	Ireland
Berlin	K. Wowereit	34	Germany
Paris	B. Delanoë	35	France

Country	Area (km²)	Population (Mio.)
England	130,395	53
Ireland	70,273	4.59
Germany	357,021	80.5
France	551,695	63.4

(b) Single-Concept Tables

Figure 6.11: Decomposition of example table based on final dependency graph.

as the determinant. Otherwise we simply select the functional dependency with the higher score to remain in the graph. After resolving bi-directionality, some of the attributes may still depend on more than one other column. Therefore, in a second step, we select the most likely candidate. Here, we follow the same procedure used to filter intra-concept dependencies. As a result, we remove coincidental as well as transitive dependencies from the graph.
In the resulting dependency graph, each entity column (except the main entity column) depends only on a single column, resembling a join graph for the concepts in the table.

Decomposition of Multi-Concept Tables
After evaluating and filtering all intra-concept and inter-concept dependencies, the remaining dependencies in the graph are considered valid deterministic functional dependencies. Based on these FDs, we decompose the multi-concept table following Algorithm 6.3.
For each entity column in the table, which represents a unique semantic concept, all *dependent* attributes, including other entity columns, are collected to form the relational representation of the concept. In our previous example, for instance, the attributes *Country*, *Area* and *Population* are grouped to form a relation representing the concept *Country*, as illustrated in Figure 6.11. In some cases, a concept is represented in a table only by its name or identifier, without additional properties, such as the concept *Mayor* in the example. Instead of forming a separate table for each of these concepts, we follow a recommendation provided by Premerlani et al. (1994) in the context of reverse engineering of databases. In the original schema, such a concept forms a *lightweight association* with another concept. Premerlani et al. suggest to simply represent these concepts as an attribute of the associated concept. Consequently, we only form a separate table, if a concept is described by more than one attribute.
By considering not only intra-concept, but also inter-concept functional dependencies when collecting all dependent attributes, we ensure that lightweight associations are represented as an attribute and that foreign key relations between concepts are preserved. After decomposing the source table into singe-concept tables, duplicate entries are removed to form valid relational tables.

Algorithm 6.3 Decomposition of relation R of a multi-concept table.

```
1:  function DECOMPOSE(R)
2:      relations ← {}
3:      for all A ∈ R do
4:          if A is entity column then
5:              relA ← {A}
6:              for all B ∈ R, A ≠ B do                ▷ Collect all dependent attributes
7:                  if A → B then
8:                      relA ← relA ∪ {B}
9:                  end if
10:             end for
11:             if |relA| > 1 then                     ▷ Check for lightweight associations
12:                 relations ← relations ∪ {relA}
13:             end if
14:         end if
15:     end for
16:     return relations
17: end function
```

6.4.5 Optimization: Table Clustering

The main reason that prevents a straightforward inference of functional dependencies from the data, is the small sample size of Web tables, which introduces a large number of spurious dependencies. In most cases, a larger sample size provides a more accurate description of the attribute domains involved in a relation. Consequently, the more samples we have to consider for the inference, the more likely it is that the inferred functional dependencies reflect meaningful semantic constraints. Therefore, in order to gain more samples for each attribute domain, we consider inferring functional dependencies collectively from multiple related tables, instead of processing each small table individually.

A similar approach is proposed by J. Wang et al. (2012), which first integrates individual relations into a single mediated schema, before extracting functional dependencies. As a result, samples from multiple sources are taken into account. However, this approach has two shortcomings with respect to Web tables: (1) Considering the scale of tables on the Web, integration into a mediated schema is not feasible. (2) Integrating data from heterogeneous and ambiguous sources is an approximate process that introduces further uncertainty (Das Sarma et al., 2009).

Therefore, we consider a trade-off between the achievable sample size and the integration effort and quality. Instead of integrating all related tables into a mediated schema, we only consider tables with equal schemas. While this constraint clearly limits the number of eligible tables, it also reduced the chance of semantic differences in the schemas and, as a result, inaccurate integration. Still, many tables on Web pages share the same schema, as illustrated in Figure 6.12. Similarly, tables with equal schemas are also common on Open Data platforms, where, for example, similar statistics are published each month or for each department.

In general, we expect a larger sample size to have a positive effect on the inference of meaningful functional dependencies. However, in some cases, clustering similar tables can decrease the accuracy of the normalization process. If the characteristics of the combined table differ significantly from the

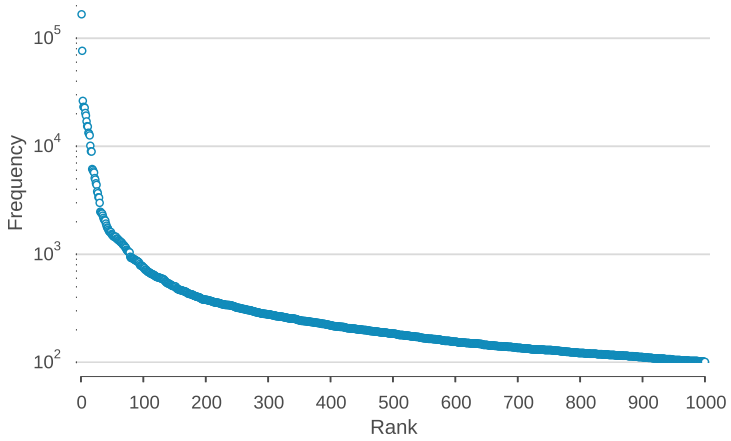


Figure 6.12: Ranked frequencies of 1000 most frequent schemas in ACSDB (Cafarella et al., 2008a).

characteristics of the individual tables, especially with respect to redundancy, evidence for valid functional dependencies can be less significant. As a result, identifying valid dependencies becomes more challenging. We illustrate these effects in more detail in the evaluation (Section 6.5.4).

6.5 EXPERIMENTAL EVALUATION

We conduct an experimental evaluation of our proposed semantic normalization approach in order to analyze the effectiveness of the method as a whole as well as of individual processing steps in particular. In our experiments, we consider real-world data that reflects the characteristics targeted by our approach.

Setup
We implemented our semantic normalization approach in Java 1.7. The tool performs an initial preprocessing of the tables provided as CSV files, the semantic normalization, as well as the evaluation of the normalization result. For various processing steps, we make use of external toolkits and software libraries. For classification and regression, we employ algorithms provided with the *WEKA* machine learning toolkit (M. Hall et al., 2009). For the lexical and linguistic analysis of attribute labels, we use the *Stanford Parser* (Socher et al., 2013).

Dataset
The dataset used in all experiments represents a collection of 100 wide tables from several data sources on the Web. To allow for a diverse set of schemas and varied data characteristics, we included data from the following sources:

- **Wikipedia**[1]**:** The Web-based free-content encyclopedia provides information in the form of articles on a wide range of topics. Many Wikipedia pages contain tables to supplement the textual

[1] `https://en.wikipedia.org`

information. The tables used in this evaluation are extracted from the English language section of Wikipedia.

- **data.gov.uk**[2]**:** The official Open Data platform of the government of the United Kingdom of Great Britain and Northern Ireland provides access to datasets released by government departments and other public sector bodies.
- **NYC Open Data**[3]**:** The official Open Data platform of the government of the City of New York offers datasets, including tabular data, published by various agencies and organizations in the city.
- **Socrata**[4]**:** The general-purpose Open Data platform by cloud software company Socrata provides centralized access to a wide range of Socrata-hosted public datasets, including government data. Publication of datasets on the platform is not restricted to official organizations, but also open to the public.

Table 6.3 highlights some statistics of the dataset. On average, the tables in the dataset contain about four individual concepts. For evaluation purposes, we also included a few single-concept tables in the corpus. However, the vast majority of tables describes multiple concepts. To illustrate the type of table considered in our evaluation, Figure 6.13 shows three of the schemas in the dataset.
The dataset includes very short tables (with only two rows of data) as well as long tables (with more than $2,500$ rows), to study the effect of sample size on the normalization quality.
All tables are manually labeled to establish a gold standard used for evaluation. The gold standard includes labels for entity columns (including main entity column), valid functional dependencies and concept affiliations. The dataset is further divided into a training set and a test set, each containing 50 tables. The training set is utilized to learn weights and train classifiers.

DBN	School	Initiative	Cohort	Principal	Email	Phone	Address	Coach

Company	Contact	Title	Phone	Fax	Address	City	State	...	Product	Function

Date	Album Name	Track	Track Title	Lyricist	Music Genre	...	Theme	Duration

Figure 6.13: Examples of schemas of multi-concept tables included in the test dataset.

Accuracy Measures
If not stated otherwise, we measure the effectiveness of our normalization approach as follows: We evaluate the accuracy at attribute level. For each attribute in a table, we regard the result as correct, if the attribute has been assigned to its respective concept. Non-entity columns must be assigned to the correct entity columns, while entity columns are either the *main* entity column or in relation with another entity column. The overall accuracy is then presented as the percentage of *correctly assigned* attributes in the table.

[2] `data.gov.uk/data/search`
[3] `https://nycopendata.socrata.com`
[4] `https://opendata.socrata.com`

Table 6.3: Statistics of tables in the test corpus.

Measure	Min	Max	Avg.
# of rows	2	2,507	187.54
# of columns	6	20	10.24
# of empty columns	0	4	0.22
# of entity columns	1	8	3.92

Table 6.4: Accuracy of the classification of entity columns.

Classifier	Accuracy (%)	
	Training	Test
Random Forest	**87.1**	**86.6**
SVM (Poly Kernel)	74.9	76.6
SVM (RBF Kernel)	85.5	85.3
Logistic Regression	74.9	77.1

6.5.1 Entity Columns

In a first experiment, we analyze and evaluate the quality of the entity column classification. In Section 6.4.1, we proposed a set of features that are suitable to identify entity columns in Web tables. In order to evaluate the general effectiveness of these features, as well as select a suitable classifier, we compare several state-of-the-art classification techniques. For the evaluation, we split the dataset into a training and a test set, each containing 50 tables. The entity column classification is performed per column. The column statistics of both sets are sufficiently balanced, with 518 columns in the training set and 484 columns in the test set. Both sets feature an entity/non-entity column ratio of roughly $2:3$.

Classifiers
In our evaluation, we consider the following classification techniques:

- **Random Forest:** As an ensemble learning technique, random forest algorithms generate multiple decision trees. The final classification result is generally the mode of the class labels returned by the individual trees. As a result, random forest algorithms are robust with respect to overfitting.
- **Support Vector Machines (SVM):** Support vector machines perform binary classification by fitting a hyperplane between the representative instances of each class, maximizing the distance

between the classes. A set of *support vectors* specifies the hyperplane, which is utilized to classify unseen instances. To perform non-linear classification, a *kernel function* can be applied, which transforms instance vectors into a higher-dimensional space that allows for a linear separation of the classes. Here, we consider a polynomial (Poly) kernel and a radial basis function (RBF) kernel. SVMs with a RBF kernel have been used by Venetis et al. (2011) to identify subject columns in single-concept tables.

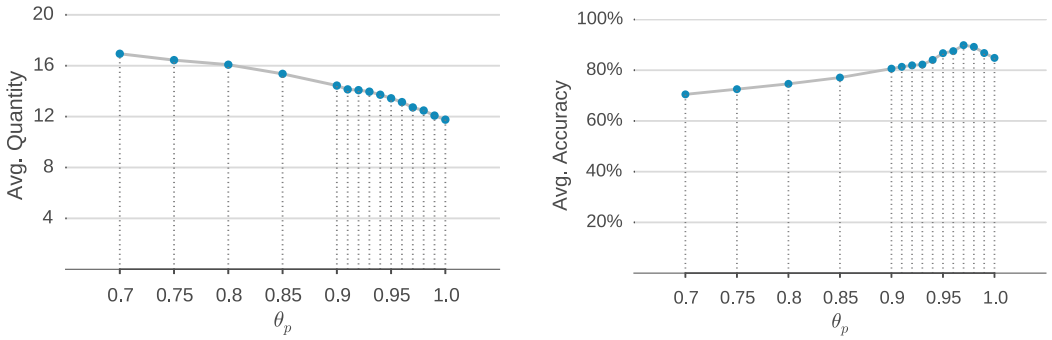
- **Logistic Regression:** As a probability model, logistic regression predicts the outcome of a binary variable from a set of features. A *logistic function* is used to model the outcome as a function of the features. Applied to classification, the result equals the probability that the considered instance belongs to the desired class.

Evaluation
For each classification technique, we identified the best parameter settings using 10-fold cross-validation on the training set. Table 6.4 shows the highest accuracy each technique achieved on the training set. Using the optimal parameters, we then evaluated each classifier on the test set. The results are also included in Table 6.4. The random forest classifier achieved the best classification result with an accuracy of 86.6%. The accuracy of support vector machines with a RBF kernel is slightly lower at 85.3%. The remaining techniques, logistic regression and SVMs with a polynomial kernel, only reach significantly lower accuracies.
Venetis et al. (2011) achieve a classification accuracy of 94% for the identification of entity columns in single-concept tables, using SVMs with a RBF kernel function. Entity columns in these tables frequently coincide with the key of the table and are generally located in the first or second column. Their characteristics are relatively homogeneous across tables and, thus, easy to classify. In contrast, entity columns in multi-concept tables exhibit much more heterogeneous characteristics. As a result, the features are less effective in characterizing entity columns. Still, the random forest classifier achieves a reasonable classification accuracy.
For all following experiments that include entity column classification, we apply the random forest classifier.

Main Entity Columns
In addition, we evaluate how well our algorithm identifies the *main* entity column amongst all entity columns in a table. As described in Section 6.4.1, we select one entity column per table to represent the main topic of the table. All other entity columns represent supplementary concepts.
Considering all entity columns in the table, our proposed selection rule identified the correct main entity column for all tables in the test set. However, if the main entity column is not identified as an entity column by the classifier, our algorithm cannot detect it at a later stage.

6.5.2 Functional Dependency Extraction and Filtering

In a second experiment, we evaluate the process of extracting and filtering functional dependencies. To separate the results from the accuracy of the entity column classifier, we use entity columns from the gold standard as input for all experiments in this section.



(a) FD Quantity

(b) Accuracy

Figure 6.14: Impact of threshold θ_p on semantic normalization.

In detail, we consider the following aspects:

- **Parameter θ_p:** We take a look at the importance of selecting the optimal threshold for probabilistic functional dependencies.
- **Weighted Scores:** We learn weights to combine scores obtained from different indicators of attribute relatedness.
- **Systematic Filtering:** We evaluate the overall effectiveness of the functional dependency extraction and filtering process.

Probabilistic Functional Dependencies

In our approach we consider probabilistic functional dependencies $X \xrightarrow{p} A$, instead of exact dependencies, to account for errors and inconsistencies in the data. Threshold parameter θ_p determines how much noise is tolerated in the data. However, as stated previously, we face a trade-off between tolerating inconsistent data and introducing spurious dependencies.

To study this trade-off, we vary parameter θ_p in our experiments in the range $[0.7, 1.0]$. Note that $\theta_p = 1.0$ corresponds to exact functional dependencies and, thus, no error tolerance. Figure 6.14 illustrates the consequences the selection of θ_p has on the extraction of functional dependencies as well as the overall normalization of tables. Both, the number of extracted dependencies as well as the normalization accuracy are presented as the average across all 50 tables in the test set.

In Figure 6.14(a), we can see that, with a lower threshold, the number of candidate dependencies we consider in our algorithm steadily increases. However, by lowering the threshold, the normalization accuracy decreases, as we extract more and more spurious functional dependencies (see Figure 6.14(b)). For our specific dataset, the optimal threshold with respect to normalization accuracy, is close to 0.97. If we set a higher threshold, we miss relevant dependencies. Yet, if we set the threshold below the optimum, we extract too many false dependencies that affect the quality of the normalization result. Overall, we obtain an increase in average accuracy of 5% by considering probabilistic functional dependencies with a threshold at $\theta_p = 0.97$, instead of expecting exact functional dependencies.

Table 6.5: Weights trained for each indicator.

Indicator	Trained Weights	
	Intra	**Inter**
Attribute Co-occurrence (ω_1)	0.0	0.5644
Common Concept Labels (ω_2)	0.3009	0.0
NULL Value Distribution (ω_3)	0.6675	0.3137
Value Correlations (ω_4)	0.2256	0.3696
Column Distance (ω_5)	0.1799	0.4465
Intercept (ω_0)	-0.4628	-0.2605

Figure 6.15 provides a detailed view of the normalization accuracy per table. Comparing the results for $\theta_p = 0.95$ and $\theta_p = 0.97$, both close to the optimal threshold, we can clearly see the compromise. Despite the small difference in threshold, we can see a significant increase in accuracy for several tables, including Tables 11, 13, 39 and 40. However, we also loose slightly in accuracy for Table 7, which is more affected by inconsistencies.
Although we can identify a clear optimal threshold for our dataset, this is not universally the case. Generally, data inconsistencies are not present to the same extend in all tables. As highlighted in our experiments, different tables require different thresholds to achieve the optimal normalization result. Ultimately, an assessment of the general quality of the tables in the dataset at hand is essential when working with probabilistic or approximate functional dependencies.

Weighting Indicator Scores
To distinguish between meaningful and spurious functional dependencies, we consider a set of five indicators to evaluate the relatedness of attributes (see Section 6.4.3). To combine individual scores into a final score, we use linear regression to learn weights from the training data, with different weights for intra-concept and inter-concept dependencies, respectively. All derived weights are shown in Table 6.5. We can see a significant difference in the weights between inter-concept and intra-concept dependencies. As expected, common concept labels have no relevance for dependencies between concepts, since they are generally shared between attributes of the same concept. Furthermore, attribute co-occurrence appears to have no relevance for intra-concept dependencies. One possible explanation is the fact that some of the attribute labels in our dataset are not included in the attribute collection utilized to derive the attribute co-occurrence score.

Dependency Evaluation and Filtering
After analyzing the impact of threshold θ_p and learning indicator weights ω_i, we now take a closer look at the effectiveness of the overall dependency evaluation and filter process. Figure 6.15 illustrates the normalization accuracy achieved per table in our test set.
For the majority of tables, we achieve an accuracy of over 80%. For 19 tables, we even associate all attributes with the correct concepts. This indicates that the scores we employ to measure attribute

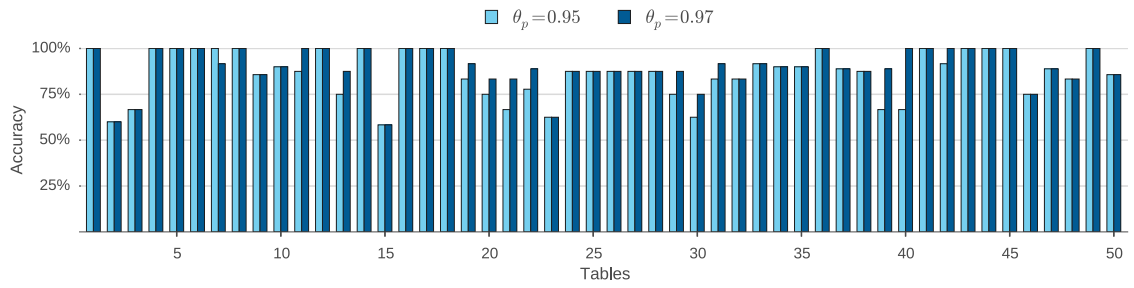


Figure 6.15: Comparison of normalization accuracy per table with respect to θ_p.

relatedness are reasonably effective in inferring meaningful functional dependencies.
However, there are a few tables that achieve a lower accuracy, including Tables $2, 3, 15$ and 23. A detailed analysis of these tables discloses the following reasons:

- **Limited Sample Size:** Table 3 contains only three rows of values, which is not sufficient to identify all meaningful correlations between attributes.
- **Constant Columns:** Columns with a single value are a challenge for our algorithm in several of these tables. As illustrated in Section 6.3, constant columns depend on any other column in the table. Consequently, any other column with a high level of redundancy appears to be highly related, based on our evaluation scores. However, in some cases such as Tables 3 and 23, this correlation is coincidental and does not reflect any meaningful semantic relation between the attributes in question.
- **Irregular Reading Order:** Assuming a logical reading order of tables from left to right, we assign higher scores to FD candidates, where the determinant is mentioned *before* the dependent (see Section 6.4.3). However, not all tables conform to this assumption. In Tables 2 and 15, for example, several attributes appear before their corresponding entity columns. If the remaining indicators do not provide sufficient evidence to identify the correct dependency, the column order and distance feature becomes the deciding factor. In these cases, our algorithm fails to normalize the table correctly.

6.5.3 End-To-End Processing

In the previous experiments, we have evaluated each processing step separately. Now, we analyze the complete process end-to-end to evaluate how errors in the identification of entity columns propagate through the normalization process and influence the accuracy of the result.
In Figure 6.16, we compare the partial evaluation of the previous experiment to the overall accuracy. We can see that the selection of entity columns has a significant effect on the final result. For several tables, we see a loss in accuracy, due to the following reasons:

- **Too Many Entity Columns:** For several tables, the classifier identifies too many entity columns, which can cause attributes to be assigned to invalid concepts. This affects primarily attributes

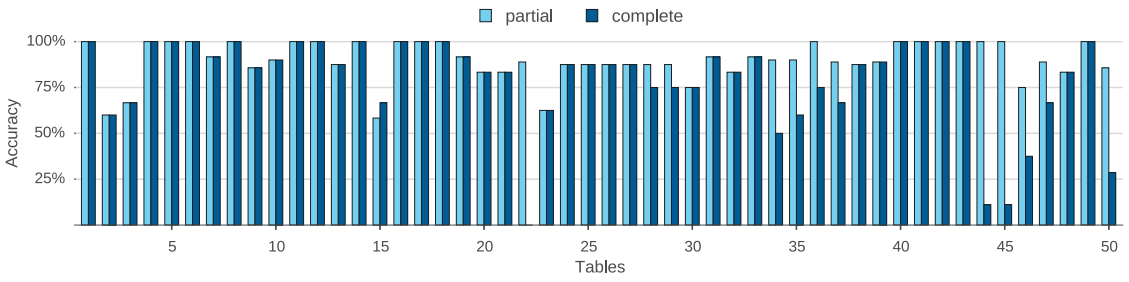


Figure 6.16: Comparison between partial and complete normalization ($\theta_p = 0.97$).

with categorical values, such as type descriptions, which have similar characteristics to valid entity columns. Examples include Tables 29, 47 and 50.

- **Too Few Entity Columns:** Missing important entity columns also has a significant impact on the normalization of a table. As the selection of candidate dependencies is driven by the set of entity columns, missing entity columns mean that all attributes of the respective concept are assigned to a different concept. Entity columns missed by the classifier either contain numeric IDs, as in Table 22, or very long string values, as in Tables 34 or 44.
- **Incorrect Main Entity Column:** In cases where the classifier cannot identify the main entity column of a table, we see the most severe impact. In our test set, we miss the main entity column for Tables 22, 44 and 45. As a result, the majority of attributes are assigned to the wrong (or no) concept.

6.5.4 Clustering

The experiments in Section 6.5.2 confirmed that the small sample size of some Web tables complicates the extraction of meaningful correlations and, ultimately, the semantic normalization of these tables. In our final experiment, we evaluate the effectiveness of table clustering (as described in Section 6.4.5) to address this challenge.
In our test set, we identified clusters for eleven different schemas, with cluster sizes between two and four tables. In total, we identified related tables for 30 tables. In Figure 6.17, we compare the normalization accuracy achieved by considering each table individually to the accuracy achieved by taking related tables into account. We can see an increase in accuracy in four different clusters. A detailed analysis of the tables involved indicates that, in all cases, the tables benefited from a reduction in *constant* columns. By considering other related columns, more distinct values were added to previously constant columns, thus reducing the number of candidate dependencies.
For two clusters, we see a lower accuracy after considering related tables. We can identify two reasons for this. First, in some cases, the correlation and redundancy between attributes that indicate a strong relatedness in individual tables, are less significant in combination with other tables. As a result, the correct dependencies are harder to detect. Second, functional dependencies that hold in individual tables might not hold across tables, which is the case in cluster C6. While the individual tables feature

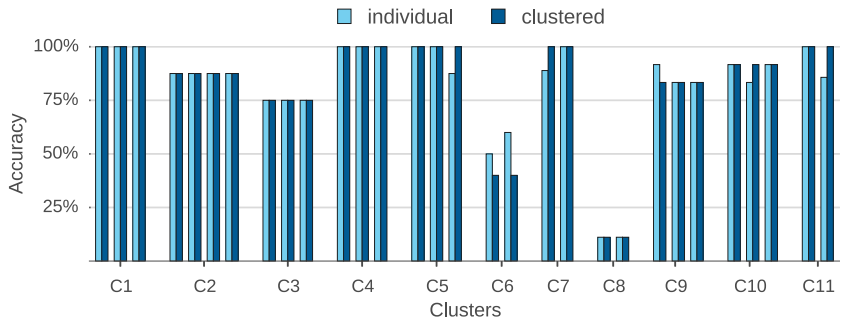


Figure 6.17: Comparison between individual and clustered tables ($\theta_p = 0.97$).

a $1 : n$ relationship between two concepts, combining both tables reveals a $n : m$ relationship, which is not detectable with simple functional dependencies.
The conducted experiment indicates mixed results regarding the effectiveness of schema clustering. This is mainly due to the limited size of the test set. First of all, we did not have related tables in the corpus for some of the smallest tables, which would have benefitted most from additional samples. Second, considering only the limited set of tables in the test corpus does not provide a good estimate of the potential of clustering tables to improve semantic normalization. From the results of our experiment, it is difficult to estimate the effect additional samples would have. It is possible that additional data could compensate some of the issues that caused a lower accuracy for some of the test tables. Additional experiments at larger scale are necessary to better evaluate the overall effectiveness. However, such experiments require immense manual effort to provide a gold standard for the normalization, and are, therefore, left for future work.

6.6 SUMMARY AND DISCUSSION

In this chapter, we addressed the simplifying *single-concept assumption* that is frequently made in connection with the interpretation and utilization of Web tables. In order to extract binary relations, identify specific concepts or identify related tables, it is assumed that a table with all its attributes characterizes only a single semantic concept. However, a detailed study of the tables available on the Web reveals that a significant amount of these tables are of a more complex nature, often describing two or more concepts and their relationships. It is clear that regarding these tables as single-concept tables results in an inaccurate interpretation and the extraction of wrong information.
Therefore, we proposed *semantic normalization*, in order to decompose complex multi-concept tables into multiple single-concept tables. In summary, we made the following contributions:

- **Web Table Characteristics:** We provide a detailed analysis of characteristic features found in tables on the Web. Especially a limited sample size, the data quality, as well as the informative value of attribute labels impact the inference of meaningful functional dependencies from the tables. Consequently, a successful inference algorithm should support uncertainty and factor in both, the correlations in the data as well as the semantics of the schema.

- **Systematic Normalization:** We approached the task of semantic normalization on Web tables by proposing a systematic extraction and evaluation process for the inference of meaningful functional dependencies. By first identifying *entity columns*, which serve as representatives (and keys) of the concepts contained in a table, we can limit the amount of dependency candidates we need to consider. As a result, we focus on inter-concept and intra-concept dependencies. Instead of *exact* functional dependencies, *probabilistic* dependencies are employed, in order to account for the varying quality and inconsistencies of Web data.

- **Indicators of Attribute Relatedness:** As part of our normalization approach, we identified several indicators that provide clues for the identification of relevant semantic relations between attributes in a table. We consider structural clues, such as relative column position, value correlations and the distribution of NULL values, as well as semantic clues, including attribute co-occurrence and common concept labels.

- **Evaluation on Real-world Data:** Finally, we conducted various experiments on a collection of tables extracted from various sources of tables on the Web to provide a realistic assessment of the effectiveness of our normalization approach. We achieved high quality results for the majority of these tables. The identification of entity columns emerged as a crucial element in the normalization process, as missing important entity columns significantly impacts the extraction and ultimate selection of functional dependencies.

Following semantic normalization, the relational schema of the tables is easier to interpret, as it is closer to the conceptual schema of the tables. In this capacity, semantic normalization is closely related to the objectives of database reverse engineering. Facilitating a better understanding of the table content ultimately improves its utilization in other applications.
While the approach we proposed achieves satisfying results for a wide range of tables, there are still opportunities for future work. A first topic targets the identification of entity columns. As this task is essential to our approach, further improving its accuracy will enhance the overall effectiveness of semantic normalization. A second topic involves an extension of our initial approach to support more complex concepts and relationships. Relevant aspects include the support for compound keys, n-ary relations, as well as $n : m$ relations between concepts. Initial analysis suggests that supporting such complex relations requires a semi-automatic process that incorporates expert support and intervention.

7

CONCLUSION

7.1 Summary of Contributions

7.2 Directions of Future Work

7.1 SUMMARY OF CONTRIBUTIONS

Web data represents an abundance of accessible information that has become an important resource for a wide range of applications. It is predominantly harnessed to satisfy multifaceted information needs, but also serves as a valuable resource for computational linguistics. In recent years, significant research effort has been directed especially at *tables* on the Web, which form a rich resource for factual and relational data, but are not natively supported by text-centric Web search applications. The objective is to recover and understand the data underlying these tables in an automated fashion that provides users and applications with better access to this information resource. Automated Web table recovery and understanding gains further importance with the growing Open Data trend that sees more and more public organizations and governmental bodies publish their data on the Web, often in tabular form.

Tables provide a compact representation of similarly structured data, with some of the meaning and intention of the content implicitly reflected by the table structure. Tables on the Web are mostly intended for human consumption, with the implicit information and structural variations relatively easy to comprehend for humans. However, processing these tables and inferring their meaning algorithmically is much more complex, and accurately extracting high quality data from Web tables becomes a challenging task. The complexity of the recovery process originates from the heterogeneity of table layouts, the semantic ambiguity of the content description, the complex dependencies between table content, structure and context, as well as the general lack of quality control on the Web.

In the literature, many important contributions have been made to implement and extend the recovery process for Web tables. However, often various *simplifying assumptions* are made about the characteristics of the tables, in order to reduce the problem space and make the complex sub-tasks involved in Web table recovery and understanding manageable. The most prominent assumptions include a uniform table layout, a single concept per table, simple concept keys or the assumption that all available context is equally relevant and related to the table content. The resulting algorithms and techniques often have a limited scope, leading to imprecise or inaccurate results when applied to tables with conflicting characteristics.

In this thesis, we addressed some of these limitations by proposing various techniques that extend the Web table recovery and, especially, understanding process in order to relax these simplifying assumptions. Our objective was to adjust different aspects of the process to better match the characteristic features of tables on the Web, thus increasing the overall scope and improving the extraction quality. For that purpose, we first investigated these tables in detail to explore their characteristics and identify the unique challenges involved in processing them. We then proposed extensions to the following areas:

- **Table Classification:** To relax the assumption of a uniform table layout, we focused on incorporating table layout classification into the table recovery process. We reviewed and consolidated classification schemes and features proposed in the literature to classify tables based on their layout. To incorporate the layout classification into the recovery process, we proposed a double-layer classification approach as an alternative to the single-layer approach proposed in the literature, which combines table detection and layout classification into a single classification problem. Our experiments showed that the double-layer approach that performs table detection and layout classification consecutively, performs as well as the previously used ap-

proach, yet provides more flexibility to adjust the feature selection and training data to the characteristics of the two separate classification problems.

- **Context and Header Recovery:** We studied the role of contextual information in Web table understanding and proposed two extensions that incorporate table context in a more focused way. First, we proposed an approach to evaluate the relevance of information in long context segments with respect to the table content, in order to reduce noise that leads to inaccurate table relevance decision, for example in table search or table matching scenarios. Our approach identifies topically coherent paragraphs in the contexts and produces a ranking and selection based on the relevance of each paragraph to the content of the table. We then utilized the table context to recover rich descriptions of table attributes by locating and extracting attribute-specific phrases in the context. Using direct and indirect matching techniques, we were able to retrieve a diverse set of contextual annotations to describe the table content. We could show that applications such as table search benefit from these annotations.

- **Conceptualization:** Conceptualization aims to identify the semantic concepts described by tables, for instance, by matching the table schema and content to entries in a knowledge base or taxonomy. Most algorithms proposed in the literature follow the single concept assumption, where all attributes in a table describe the same semantic context, to reduce the complexity of the matching task. In order to relax this assumption, we proposed a semantic normalization technique that decomposes multi-concept tables into single-concept tables, which can then be processed as before. The algorithm specifically targets Web tables, taking into account missing or non-informative attribute labels, inconsistencies in the data as well as affects caused by the often small size of Web tables.

We addressed several aspects of Web table understanding in order to extend existing approaches to cover a wider range of tables and to increase the overall accuracy. These extensions target various important requirements of Web table applications, such as ontology learning or question answering, including the need for descriptive attribute labels, the identification of semantic concepts in the tables as well as recovery of functional dependencies that hold in the data.

7.2 DIRECTIONS OF FUTURE WORK

Although we were able to extend and improve the Web table understanding process in several directions, there are still many challenges and open issues that need to be addressed before the potential of Web tables and the information contained in them can be fully harnessed. Web table recovery and, especially, Web table understanding represent very challenging tasks due to the heterogeneity and ambiguity of the sources. For many subtasks involved, there is still room to improve especially the precision of the algorithms. Furthermore, we require additional techniques for instance, to identify temporal or spatial dimension in the tables, to extract specific constraints such as units or hidden attributes from the context, or to identify compound keys for complex concepts in the tables. So, although significant progress has been made in recent years to automatically recover facts and relations from Web tables, this research area is far from being completely covered.

To conclude this thesis, we would like to highlight two important challenges that we consider essential in order to further advance Web table understanding research.

1. **Standardized Test Data:** An issue that we faced in this thesis, and which has been pointed out by other researchers, is the general lack of standardized test data for Web table understanding, which leads to repeated data collection efforts and at the same time prevents a meaningful and fair comparison of competing solutions.
   Many of the sub-tasks involved in Web table understanding require manually labeled test data to evaluate their suitability and performance. For each task that is studied, a representative set of tables must be collected and labeled by human judges, which often requires a lot of time and effort. Standardized public test corpora could significantly reduce this effort, by eliminating the need for repeated extraction and labeling. Furthermore, standardization of test sets would ensure that comparisons between alternative approaches are fair. The Web is constantly evolving and even on Wikipedia tables and other content change frequently. Therefore, consistent and stable corpora are required to enable the replication and comparison of results.
   So far, a number of Web table corpora have been published, including the *Web Data Commons - Web Tables*[1] collection and the *Dresden Web Table Corpus*[2], yet without specific labels for table understanding tasks. A first effort of a labeled data collection is the *T2D Gold Standard*[3] for the evaluation of systems that match tables to the DBPedia knowledge base.

2. **Human Involvement and Support:** The second challenge addresses the quality of table understanding. Many of the sub-tasks involved in the table understanding process , such as identifying related tables, are easy to comprehend for humans, yet it is very challenging to achieve the same accuracy algorithmically. It is well established that schema matching and integration of heterogeneous and ambiguous tables inherently requires human assistance to produce high quality results (see, for instance, Halevy (2005)). However, the sheer volume of a Web table corpus limits the amount of work that can be performed by humans, whether they are domain experts or the crowd. Therefore, hybrid techniques that harness and combine the strengths of both, human and machine computation, effectively, are the most sensible solutions. First proposals, targeting the matching of attribute labels to knowledge base entries (Fan et al., 2014) or the extraction of named entities from the table context (Braunschweig et al., 2013) show promising results. However, in the long run, a holistic approach to incorporate humans is required that does not target a specific task, but enables the utilization of user interaction and feedback in all aspects of table understanding, if necessary. Many tasks, such as the identification of compound keys or the recovery of $n : m$ relations between concepts in a table, can benefit from human support.

[1] `http://webdatacommons.org/webtables/`
[2] `https://wwwdb.inf.tu-dresden.de/misc/dwtc/`
[3] `http://webdatacommons.org/webtables/goldstandard.html`

A

LAYOUTS OF TABLES ON THE WEB

A.1 Vertical Listing

A.2 Horizontal Listing

A.3 Matrix

A.4 Special Cases

A.1 VERTICAL LISTING

Rank	Island	Area (sq mi)	Area (km²)
1	Isle of Wight	147.09	380.99
2	Isle of Sheppey	36.31	94.04
3	Hayling Island	10.36	26.84
4	Foulness Island	10.09	26.84
5	Portsea Island	9.36	24.25
6	Canvey Island	7.12	18.45
7	Mersea Island	6.96	18.04
8	Walney Island	5.01	12.99
9	Wallasea Island	4.11	10.65

Figure A.1: Example of table in category *ExA* [1].

Historical population		
Census	Pop.	%±
1950	782	—
1960	382	–51.2%
1970	370	–3.1%
1980	5,949	1,507.8%
1990	8,386	41.0%
2000	10,395	24.0%
2010	17,056	64.1%

Figure A.2: Example of table in category *DxA* [2].

A.2 HORIZONTAL LISTING

2010 Census Data	New York City	Los Angeles	Chicago
Total population	8,175,133	3,792,820	2,695,598
Population, percent change, 2000 to 2010	+2.1%	+2.6%	-6.9%
Population density	27,012 /sq. mi.	8,092 /sq. mi.	11,864 /sq. mi.

Figure A.3: Example of table in category *AxE* [3].

Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Average high °F (°C)	38 (3)	41 (5)	50 (10)	61 (16)	71 (22)	80 (27)	85 (29)	83 (28)	75 (24)	65 (18)	54 (12)	43 (6)
Average low °F (°C)	18 (–8)	19 (–7)	27 (–3)	36 (2)	46 (8)	54 (12)	59 (15)	58 (14)	51 (11)	39 (4)	32 (0)	23 (–5)
Precipitation inches (mm)	4.50 (114.3)	3.00 (76.2)	4.41 (112)	4.64 (117.9)	5.09 (129.3)	4.40 (111.8)	5.29 (134.4)	4.37 (111)	5.33 (135.4)	4.17 (105.9)	4.37 (111)	4.10 (104.1)

Figure A.4: Example of table in category *AxD* [4].

[1] Source: `https://en.wikipedia.org/wiki/List_of_islands_of_England`
[2] Source: `https://en.wikipedia.org/wiki/Tribeca`
[3] Source: `https://en.wikipedia.org/wiki/New_York_City`
[4] Source: `https://en.wikipedia.org/wiki/New_York_metropolitan_area`

A.3 MATRIX

GDP per capita (current USD)

	2008	2009	2010	2011	2012
United States of America	46,760	45,305	46,612	48,112	49,641
United Kingdom	43,147	35,331	36,238	38,974	39,090

Figure A.5: Example of table in category *ExD* [5].

Racial composition [hide]	2012[1]	1990[151]	1950[151]	1900[151]
White	65.2%	58.3%	79.4%	97.8%
—Non-Hispanic	47.6%	48.9%	n/a	n/a
Black or African American	18.4%	22.0%	19.6%	2.0%
Hispanic or Latino (of any race)	25.8%	26.0%	n/a	n/a
Asian	12.0%	7.4%	0.8%	0.3%

Figure A.6: Example of table in category *DxD* [6].

A.4 SPECIAL CASES

Government[3]	
• **Type**	Mayor–Council
• **Body**	New York City Council
• **Mayor**	Bill de Blasio (D)
Area[2]	
• **Total**	468.9 sq mi (1,214 km²)
• **Land**	304.8 sq mi (789 km²)
• **Water**	164.1 sq mi (425 km²)
• **Metro**	13,318 sq mi (34,490 km²)
Elevation[4]	33 ft (10 m)

Figure A.7: Example of Attribute/Value table [6].

New York City's five boroughs overview				
Jurisdiction		Population	Land area	
Borough	*County*	*1 July 2013 Estimates*	*square miles*	*square km*
Manhattan	**New York**	**1,626,159**	**23**	**59**
The Bronx	Bronx	1,418,733	42	109
Brooklyn	Kings	2,592,149	71	183
Queens	Queens	2,296,175	109	283
Staten Island	Richmond	472,621	58	151
City of New York		**8,405,837**	**303**	**786**
State of New York		19,651,127	47,214	122,284
Source: United States Census Bureau[44][45][46]				

Figure A.8: Example of a complex nested table [6].

[5] Source: `https://en.wikipedia.org/wiki/Gross_domestic_product`
[6] Source: `https://en.wikipedia.org/wiki/New_York_City`

B

CORPORA

B.1 Dresden Web Table Corpus

B.2 Wikipedia Corpus

B.1 DRESDEN WEB TABLE CORPUS

The *Dresden Web Table Corpus*[1] *(DWTC)* is a large corpus consisting of roughly 145 million tables extracted from HTML pages. The corpus was extracted from the July 2014 version of the *Common Crawl*[2], a publicly available archive of crawled Web pages, published by the Common Crawl Foundation, which contains 3.6 billion Web pages.
The table detection and classification process described in Chapter 4 was used to identify genuine tables and their respective layout categories. Figure B.1 shows the distribution of identified layout types in the corpus. Vertical and horizontal listings make up the majority of tables in the Web, with only a very small percentage of the tables falling into the matrix category. Table B.1 presents the table size statistics for the corpus. Although very large tables exist in the corpus, with up to 46,743 rows and 113,682 columns, the average table size is much smaller, confirming the assumption that most tables on the Web are small compared to tables in enterprise database systems. The average column size for tables in the category *Horizontal Listings* is 2.47, indicating a high frequency of attribute/value tables, which typically feature 2 columns.

Table B.1: Size statistics for tables in the DWTC, organized by layout category.

Layout	Rows			Columns		
	Min	Max	Avg.	Min	Max	Avg.
· Vertical Listing	2	28,891	17.16	2	7,291	5.79
· Horizontal Listing	2	46,743	8.96	2	6,878	2.47
· Matrix	2	9,030	17.69	3	2,035	7.50
· All	2	46,743	12.70	2	113,682	4.16

B.2 WIKIPEDIA CORPUS

The Wikipedia corpus represents a subset of the DWTC, limited to tables extracted from the English Wikipedia. The corpus makes up about 1% of all tables in the DWTC. As illustrated in Figure B.2, the distribution of layout categories is very similar to the overall distribution in the DWTC. The category *Horizontal Listing* is slightly more dominant, which can be explained by the frequent use of fact sheets (i.e. attribute/value tables) in Wikipedia articles to describe entities such as cities, countries or people of public interest. Again, matrix tables are very rare in the corpus.
Table B.1 shows the table size statistics of the corpus. The average table size, regarding row as well as column size, is consistent with the average table size reported for the DWTC. However, there less outliers with very large column and row sizes in the corpus, which is attributed to the higher overall quality of Wikipedia content, compared to the Web in general.

[1] `https://wwwdb.inf.tu-dresden.de/misc/dwtc/`
[2] `https://commoncrawl.org`

Table B.2: Size statistics for tables in the Wikipedia corpus, organized by layout category.

Layout	Rows			Columns		
	Min	Max	Avg.	Min	Max	Avg.
• Vertical Listing	2	5,055	14.41	2	96	5.29
• Horizontal Listing	2	2,453	10.49	2	47	2.07
• Matrix	2	4,312	4.59	3	52	8.12
• All	2	5,055	11.87	2	96	3.36

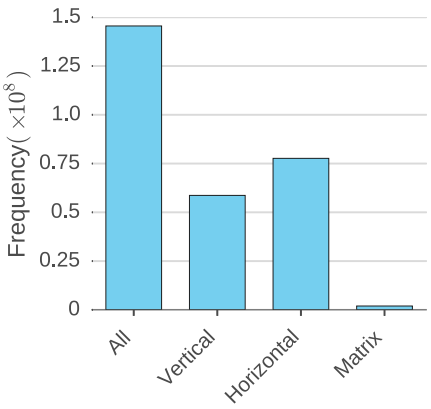


Figure B.1: Distribution of layout categories in DWTC.

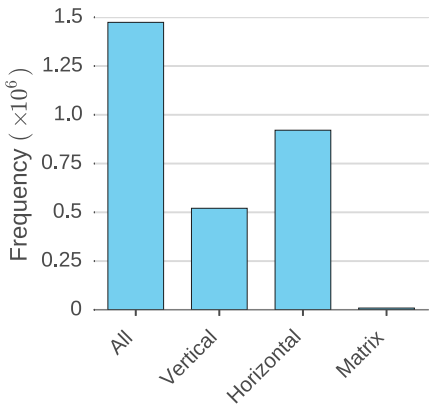


Figure B.2: Distribution of layout categories in the Wikipedia Corpus.

C

WEB TABLE CONTEXT

C.1 Embedded Web Tables

C.2 Open Data Tables

In this section, we present examples of resources for context information found on the Web. We distinguish between Tables embedded in HTML and tables published on Open Data platforms.

C.1 EMBEDDED WEB TABLES

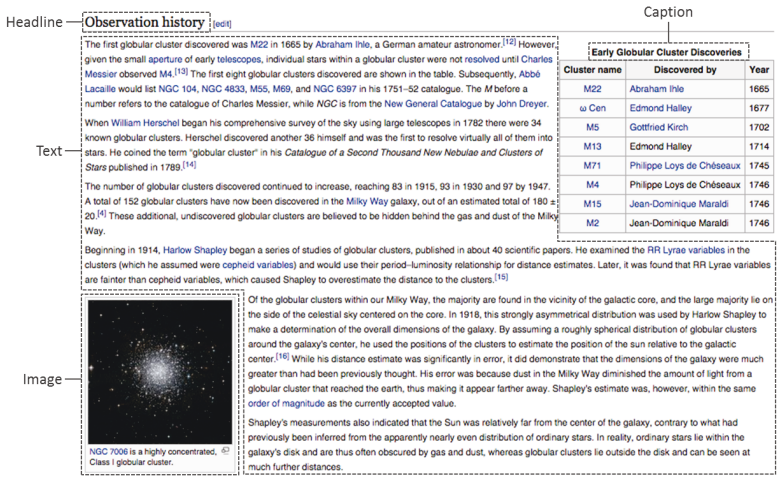


Figure C.1: Selection of contextual information provided for a table embedded in HTML [1].

[1] Source: `https://en.wikipedia.org/wiki/Globular_cluster`

C.2 OPEN DATA TABLES

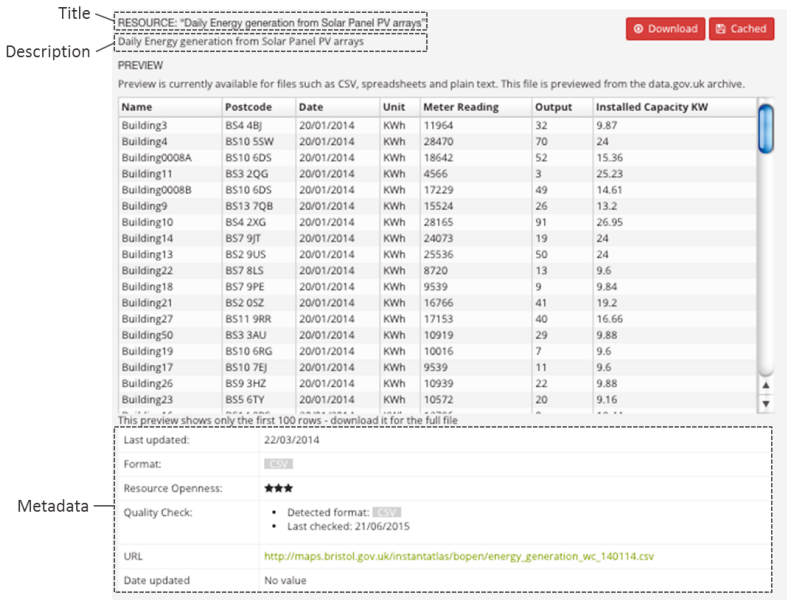


Figure C.2: Selection of contextual information provided for a resource published on an Open Data platform [2].

[2] Source: `http://data.gov.uk` Resource: `d0dcb147-ab61-45ba-b28f-8c376f52d66a`

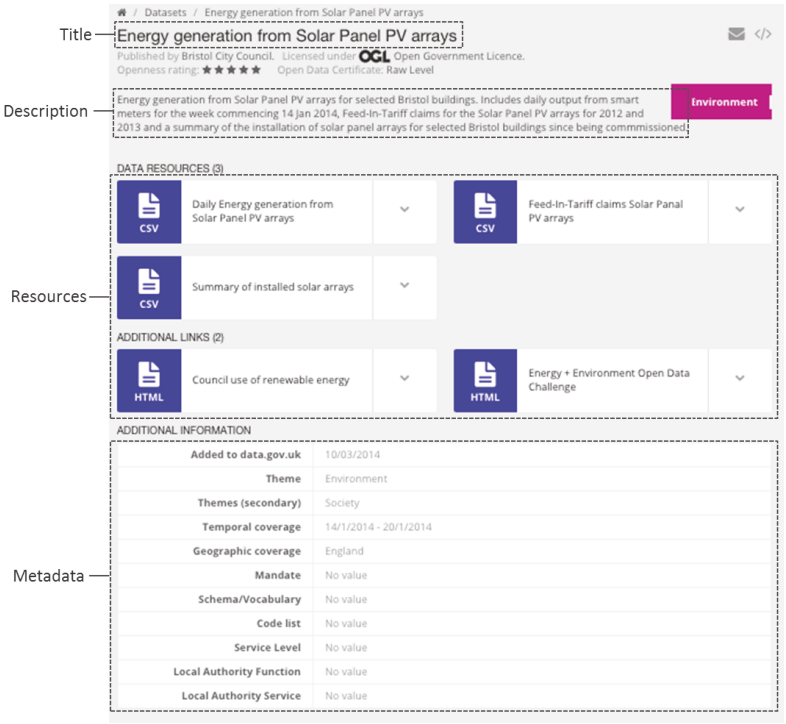


Figure C.3: Selection of contextual information provided for a dataset published on an Open Data platform [3].

[3] Source: `http://data.gov.uk` Dataset: "Energy Generation from Solar Panel PV Arrays"

D

EVALUATION OF CONTEXT RELEVANCE

D.1 Retrieval Functions

D.2 Text Similarity Measures

D.3 Topic Models

In this section, we include the detailed experimental results of the context relevance estimation. For different similarity measures, we evaluate the mean reciprocal rank (MRR) and the mean average precision (MAP). Furthermore, we distinguish between small, medium sized and large tables.

D.1 RETRIEVAL FUNCTIONS

Table D.1: Detailed evaluation of retrieval functions to estimate context relevance. The scores represent the mean reciprocal rank (MRR), distinguishing between small, medium and large tables based on the number of terms in the table.

Retrieval Function	$\mathbf{< 20}$	$\mathbf{20 < x < 200}$	$\mathbf{> 200}$	**All**
Vector Space Model				
· TF	1.0	0.924	1.0	0.944
· TF-IPF	1.0	0.947	1.0	0.961
· TF-IDF (Documents)	1.0	0.933	1.0	0.951
· TF-IDF (Passages)	1.0	0.932	1.0	0.950
Language Models				
· Dirichlet Smoothing	1.0	0.969	1.0	0.978
· Jelinek-Mercer Smoothing	1.0	0.947	1.0	0.961
Okapi BM25 (short queries)				
· No IDF	1.0	0.969	1.0	0.977
· IPF	1.0	0.969	1.0	0.977
· IDF (Documents)	1.0	0.911	1.0	0.934
· IDF (Passages)	1.0	0.943	1.0	0.958
Okapi BM25 (long queries)				
· No IDF	1.0	0.969	0.916	0.961
· IPF	1.0	0.969	0.916	0.961
· IDF (Documents)	1.0	0.933	1.0	0.951
· IDF (Passages)	1.0	0.943	0.916	0.942

Table D.2: Detailed evaluation of retrieval functions to estimate context relevance. The scores represent the mean average precision (MAP).

Retrieval Function	$<$ **20**	**20** $<$ **x** $<$ **200**	$>$ **200**	**All**
Vector Space Model				
· TF	0.916	0.787	0.887	0.816
· TF-IPF	0.875	0.811	0.881	0.829
· TF-IDF (Documents)	0.806	0.807	0.904	0.827
· TF-IDF (Passages)	0.806	0.818	0.928	0.839
Language Models				
· Dirichlet Smoothing	0.821	0.857	0.912	0.865
· Jelinek-Mercer Smoothing	0.821	0.828	0.910	0.844
Okapi BM25 (short queries)				
· No IDF	0.821	0.813	0.890	0.829
· IPF	0.833	0.812	0.878	0.827
· IDF (Documents)	0.806	0.796	0.951	0.827
· IDF (Passages)	0.806	0.807	0.909	0.827
Okapi BM25 (long queries)				
· No IDF	0.917	0.817	0.881	0.837
· IPF	0.875	0.821	0.872	0.835
· IDF (Documents)	0.813	0.817	0.951	0.844
· IDF (Passages)	0.813	0.811	0.907	0.831

D.2 TEXT SIMILARITY MEASURES

Table D.3: Detailed evaluation of text similarity measures to estimate context relevance. The scores represent the mean reciprocal rank (MRR), distinguishing between small, medium and large tables based on the number of terms in the table.

Similarity Measure	$\mathbf{< 20}$	$\mathbf{20 < x < 200}$	$\mathbf{> 200}$	**All**
Cosine Similarity				
• TF	1.0	0.969	1.0	0.977
• TF-IPF	1.0	0.947	1.0	0.961
• TF-IDF (Documents)	1.0	0.924	1.0	0.944
• TF-IDF (Passages)	1.0	0.913	1.0	0.936
Language Models				
• Dirichlet Smoothing	1.0	0.969	1.0	0.978
• Jelinek-Mercer Smoothing	1.0	0.947	1.0	0.961
Okapi BM25				
• No IDF	1.0	0.924	0.916	0.928
• IPF	1.0	0.954	0.916	0.950
• IDF (Documents)	1.0	0.935	0.888	0.930
• IDF (Passages)	1.0	0.902	1.0	0.928

Table D.4: Detailed evaluation of text similarity measures to estimate context relevance. The scores represent the mean average precision (MAP).

Similarity Measure	$<$ **20**	**20** $<$ **x** $<$ **200**	$>$ **200**	**All**
Cosine Similarity				
• TF	0.875	0.850	0.947	0.871
• TF-IPF	0.875	0.811	0.948	0.843
• TF-IDF (Documents)	0.792	0.825	0.965	0.850
• TF-IDF (Passages)	0.792	0.826	0.977	0.854
Language Models				
• Dirichlet Smoothing	0.875	0.846	0.940	0.867
• Jelinek-Mercer Smoothing	0.800	0.854	0.901	0.859
Okapi BM25				
• No IDF	0.821	0.814	0.819	0.816
• IPF	0.850	0.816	0.816	0.819
• IDF (Documents)	0.813	0.818	0.931	0.840
• IDF (Passages)	0.806	0.793	0.891	0.813

D.3 TOPIC MODELS

Table D.5: Detailed evaluation of LDA-based similarity measures to estimate context relevance. The scores represent the mean reciprocal rank (MRR), distinguishing between small, medium and large tables based on the number of terms in the table.

Settings	$<$ **20**	**20** $<$ **x** $<$ **200**	$>$ **200**	**All**
$\mathbf{sim_H}$				
· $K = 200$	0.536	0.541	0.583	0.549
· $K = 500$	0.600	0.502	0.806	0.569
· $K = 1,000$	0.250	0.519	0.700	0.538
$\mathbf{sim_{KL}}$				
· $K = 200$	0.541	0.569	0.583	0.569
· $K = 500$	0.350	0.506	0.778	0.549
· $K = 1,00$	0.250	0.528	0.755	0.555

Table D.6: Detailed evaluation of LDA-based similarity measures to estimate context relevance. The scores represent the mean average precision (MAP).

Settings	$\mathbf{< 20}$	$\mathbf{20 < x < 200}$	$\mathbf{> 200}$	**All**
$\mathbf{sim_H}$				
· $K = 200$	0.306	0.461	0.519	0.462
· $K = 500$	0.373	0.393	0.617	0.437
· $K = 1,000$	0.194	0.444	0.592	0.457
$\mathbf{sim_{KL}}$				
· $K = 200$	0.313	0.489	0.527	0.485
· $K = 500$	0.246	0.407	0.588	0.433
· $K = 1,00$	0.191	0.430	0.606	0.449

BIBLIOGRAPHY

Abadi, Daniel et al. (2014). "The Beckman Report on Database Research". In: *SIGMOD Record* 43.3, pp. 61–70.

Abiteboul, Serge, Richard Hull, and Victor Vianu, eds. (1995). *Foundations of Databases: The Logical Level*. 1st Edition. Addison-Wesley Longman Publishing Co., Inc.

Adelfio, Marco D. and Hanan Samet (2013). "Schema Extraction for Tabular Data on the Web". In: *Proceedings of the VLDB Endowment* 6.6, pp. 421–432.

Allan, James (2002). "Introduction to Topic Detection and Tracking". In: *Topic Detection and Tracking*. Kluwer Academic Publishers, pp. 1–16.

Armstrong, William W. (1974). "Dependency Structures of Data Base Relationships". In: *Proceedings of the IFIP Congress*, pp. 580–583.

Ashburner, Michael et al. (2000). "Gene Ontology: Tool for the Unification of Biology". In: *Nature Genetics* 25.1, pp. 25–29.

Bahmani, Amir H., Mahmoud Naghibzadeh, and Behnam Bahmani (2008). "Automatic Database Normalization and Primary Key Generation". In: *Proceedings of the 21st Canadian Conference on Electrical and Computer Engineering*, pp. 11–16.

Bahmani, Amir H., S Kazem Shekofteh, Mahmoud Naghibzadeh, and Hossein Deldari (2010). "Parallel Algorithms for Automatic Database Normalization". In: *Proceedings of the 2nd International Conference on Computer and Automation Engineering*. Vol. 2. IEEE, pp. 157–161.

Balakrishnan, Sreeram et al. (2015). "Applying WebTables in Practice". In: *Seventh Biennial Conference on Innovative Data Systems Research*.

Beeferman, Doug, Adam Berger, and John Lafferty (1999). "Statistical Models for Text Segmentation". In: *Machine Learning - Special Issue on Natural Language Learning* 34.1-3, pp. 177–210.

Bernstein, Philip A. (1976). "Synthesizing Third Normal Form Relations from Functional Dependencies". In: *ACM Transactions on Database Systems* 1.4, pp. 277–298.

Blei, David M. and John D. Lafferty (2009). "Topic Models". In: *Text Mining: Classification, Clustering, and Applications* 10, p. 71.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). "Latent Dirichlet Allocation". In: *The Journal of Machine Learning Research* 3, pp. 993–1022.

Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik (1992). "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the 5th Annual ACM Conference on Computational Learning Theory*, pp. 144–152.

Braunschweig, Katrin, Julian Eberius, Maik Thiele, and Wolfgang Lehner (2012). "The State of Open Data - Limits of Current Open Data Platforms". In: *Proceedings of the Web Science Track at the 21st World Wide Web Conference*.

Braunschweig, Katrin, Maik Thiele, Julian Eberius, and Wolfgang Lehner (2013). "Enhancing Named Entity Extraction by Effectively Incorporating the Crowd". In: *Workshopband der 15. GI-Fachtagung für Datenbanksysteme in Business, Technology und Web*, pp. 181–195.

Braunschweig, Katrin, Maik Thiele, Julian Eberius, and Wolfgang Lehner (2015a). "Column-specific Context Extraction for Web Tables". In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pp. 1072–1077.

Braunschweig, Katrin, Maik Thiele, and Wolfgang Lehner (2015b). "From Web Tables to Concepts: a Semantic Normalization Approach". In: *Proceedings of the 34th International Conference on Conceptual Modeling*, pp. 247–260.

Breiman, Leo (2001). "Random Forests". In: *Machine Learning* 45.1, pp. 5–32.

Breiman, Leo, Jerome Friedman, R Olshen, and Charles Stone (1984). *Classification and Regression Trees*. Wadsworth.

Cafarella, Michael J. and Alon Y. Halevy (2011a). "Web Data Management". In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pp. 1199–1200.

Cafarella, Michael J., Alon Y. Halevy, and Nodira Khoussainova (2009). "Data Integration for the Relational Web". In: *Proceedings of the VLDB Endowment* 2 (1), pp. 1090–1101.

Cafarella, Michael J., Alon Y. Halevy, and Jayant Madhavan (2011b). "Structured Data on the Web". In: *Communications of the ACM* 54.2, pp. 72–79.

Cafarella, Michael J., Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang (2008a). "WebTables: Exploring the Power of Tables on the Web". In: *Proceedings of the VLDB Endowment* 1.1, pp. 538–549.

Cafarella, Michael J., Alon Y. Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu (2008b). "Uncovering the Relational Web". In: *Proceedings of the 11th International Workshop on the Web and Databases: In Conjunction with the 2008 ACM SIGMOD International Conference on Management of Data*.

Cesarini, Francesca, Simone Marinai, L. Sarti, and Giovanni Soda (2002). "Trainable Table Location in Document Images". In: *Proceedings of the 16th International Conference on Pattern Recognition*. Vol. 3. IEEE, pp. 236–240.

Chen, Hsin-Hsi, Shih-Chung Tsai, and Jin-He Tsai (2000). "Mining Tables from Large Scale HTML Texts". In: *Proceedings of the 18th Conference on Computational Linguistics*. Vol. 1. ACL, pp. 166–172.

Chen, Zhe and Michael J. Cafarella (2013). "Automatic Web Spreadsheet Data Extraction". In: *Proceedings of the 3rd International Workshop on Semantic Search Over the Web*. ACM, 1:1–1:8.

Chen, Zhe and Michael J. Cafarella (2014). "Integrating Spreadsheet Data via Accurate and Low-effort Extraction". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 1126–1135.

Chiang, Roger H. L., Terence M. Barron, and Veda C. Storey (1994). "Reverse Engineering of Relational Databases: Extraction of an EER Model from a Relational Database". In: *Data and Knowledge Engineering* 12.2, pp. 107–142.

Choi, Freddy Y. Y. (2000). "Advances in Domain Independent Linear Text Segmentation". In: *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*. ACL, pp. 26–33.

Codd, Edgar F. (1971). "Further Normalization of the Data Base Relational Model". In: *IBM Research Report* RJ909.

Codd, Edgar F. Codd F. (1970). "A Relational Model of Data for Large Shared Data Banks". In: *Communications of the ACM* 13.6, pp. 377–387.

Crestan, Eric and Patrick Pantel (2011). "Web-scale Table Census and Classification". In: *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*. ACM, pp. 545–554.

Das Sarma, Anish, Xin Luna Dong, and Alon Y. Halevy (2009). "Data Modeling in Dataspace Support Platforms". In: *Conceptual Modeling: Foundations and Applications*. Vol. 5600. Springer Berlin Heidelberg, pp. 122–138.

Das Sarma, Anish, Lujun Fang, Nitin Gupta, Alon Y. Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu (2012). "Finding Related Tables". In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 817–828.

Dechter, R. (1987). "Decomposing an N-ary Relation into a Tree of Binary Relations". In: *Proceedings of the 6th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM, pp. 185–189.

Deerwester, Scott C., Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman (1990). "Indexing by Latent Semantic Analysis". In: *Journal of the American Society for Information Science* 41.6, pp. 391–407.

Delobel, Claude and Richard G. Casey (1973). "Decomposition of a Data Base and the Theory of Boolean Switching Functions". In: *IBM Journal of Research and Development* 17.5, pp. 374–386.

Dong, Xin, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang (2014). "Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 601–610.

Eberius, Julian, Maik Thiele, Katrin Braunschweig, and Wolfgang Lehner (2012). "DrillBeyond: Enabling Business Analysts to Explore the Web of Open Data". In: *Proceedings of the VLDB Endowment* 5.12, pp. 1978–1981.

Embley, David W., Douglas M. Campbell, Y. S. Jiang, Stephen W. Liddle, Deryle W. Lonsdale, Y.—K. Ng, and Randy D. Smith (1999). "Conceptual-model-based Data Extraction from Multiple-record Web Pages". In: *Data and Knowledge Engineering* 31.3, pp. 227–251.

Embley, David W., Matthew Hurst, Daniel Lopresti, and George Nagy (2006). "Table-processing Paradigms: a Research Survey". In: *International Journal of Document Analysis and Recognition* 8.2-3, pp. 66–86.

Embley, David W., Cui Tao, and Stephen W. Liddle (2005). "Automating the Extraction of Data from HTML Tables with Unknown Structure". In: *Data and Knowledge Engineering* 54.1, pp. 3–28.

Fan, Ju, Meiyu Lu, Beng Chin Ooi, Wang-Chiew Tan, and Meihui Zhang (2014). "A Hybrid Machine-crowdsourcing System for Matching Web Tables". In: *Proceedings of the 30th International Conference on Data Engineering*. IEEE, pp. 976–987.

Franklin, Michael J., Alon Y. Halevy, and David Maier (2005). "From Databases to Dataspaces: a New Abstraction for Information Management". In: *SIGMOD Record* 34 (4), pp. 27–33.

Gatterbauer, Wolfgang, Paul Bohunsky, Marcus Herzog, Bernhard Krüpl, and Bernhard Pollak (2007). "Towards Domain-independent Information Extraction from Web Tables". In: *Proceedings of the 16th International Conference on World Wide Web*. ACM, pp. 71–80.

Göbel, Max, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi (2012). "A Methodology for Evaluating Algorithms for Table Understanding in PDF Documents". In: *Proceedings of the 2012 ACM Symposium on Document Engineering*. ACM, pp. 45–48.

Gray, Jim, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh (1997). "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals". In: *Data Mining and Knowledge Discovery* 1.1, pp. 29–53.

Halevy, Alon Y. (2004). "Structures, Semantics and Statistics". In: *Proceedings of the 13th International Conference on Very Large Data Bases*. Vol. 30. VLDB Endowment, pp. 4–6.

Halevy, Alon Y. (2005). "Why Your Data Won't Mix". In: *Queue - Semi-structured Data* 3.8, pp. 50–58.

Halevy, Alon Y., Michael J. Franklin, and David Maier (2006). "Principles of Dataspace Systems". In: *Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. ACM, pp. 1–9.

Halevy, Alon Y., Peter Norvig, and Fernando Pereira (2009). "The Unreasonable Effectiveness of Data". In: *IEEE Intelligent Systems* 24.2, pp. 8–12.

Hall, Mark A (1999). "Correlation-based Feature Selection for Machine Learning". PhD thesis. The University of Waikato.

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten (2009). "The WEKA Data Mining Software: an Update". In: *ACM SIGKDD Explorations Newsletter* 11.1, pp. 10–18.

Hearst, Marti A. (1997). "TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages". In: *Computational Linguistics* 23.1, pp. 33–64.

Hirschman, Lynette and Robert Gaizauskas (2001). "Natural Language Question Answering: The View from Here". In: *Natural Language Engineering* 7.4, pp. 275–300.

Hofmann, Thomas (1999). "Probabilistic Latent Semantic Indexing". In: *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pp. 50–57.

Huddleston, Rodney D. and Geoffrey K. Pullum (2002). *The Cambridge Grammar of the English Language*. Cambridge University Press.

Huhtala, Ykä, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen (1999). "TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies." In: *Computer Journal* 42.2, pp. 100–111.

Hurst, Matthew (2000). "The Interpretation of Tables in Texts". PhD thesis. University of Edinburgh.

Hurst, Matthew (2003). "A Constraint-based Approach to Table Structure Derivation". In: *Proceedings of the 7th International Conference on Document Analysis and Recognition*. Vol. 2. IEEE, p. 911.

Ilyas, Ihab F., Volker Markl, Peter Haas, Paul Brown, and Ashraf Aboulnaga (2004). "CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies". In: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 647–658.

Kilgarriff, Adam and Gregory Grefenstette (2003). "Introduction to the Special Issue on the Web As Corpus". In: *Computational Linguistics* 29.3, pp. 333–347.

Kurland, Oren (2006). "Inter-document Similarities, Language Models, and Ad Hoc Information Retrieval". PhD thesis. Cornell University.

Lautert, Larissa R., Marcelo M. Scheidt, and Carina F. Dorneles (2013). "Web Table Taxonomy and Formalization". In: *SIGMOD Record* 42.3, pp. 28–33.

Limaye, Girija, Sunita Sarawagi, and Soumen Chakrabarti (2010). "Annotating and Searching Web Tables using Entities, Types and Relationships". In: *Proceedings of the VLDB Endowment* 3 (1-2), pp. 1338–1347.

Ling, Xiao, Alon Y. Halevy, Fei Wu, and Cong Yu (2013). "Synthesizing Union Tables from the Web". In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 2677–2683.

Liu, Ling and M. Tamer Özsu, eds. (2009). *Encyclopedia of Database Systems*. Springer Publishing Company, Inc.

Lopresti, Daniel and George Nagy (1999). "Automated Table Processing: An (Opinionated) Survey". In: *Proceedings of the 3rd IAPR International Workshop on Graphics Recognition*, pp. 109–134.

Lopresti, Daniel and George Nagy (2000). "A Tabular Survey of Automated Table Processing". In: *Selected Papers from the 3rd International Workshop on Graphics Recognition, Recent Advances*, pp. 93–120.

Mannila, Heikki and Kari-Jouko Räihä (1992). "On the Complexity of Inferring Functional Dependencies". In: *Discrete Applied Mathematics* 40.2, pp. 237–243.

Mannila, Heikki and Kari-Jouko Räihä (1994). "Algorithms for Inferring Functional Dependencies from Relations". In: *Data and Knowledge Engineering* 12.1, pp. 83–99.

Miller, George A. (1995). "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11, pp. 39–41.

Mulwad, Varish, Tim Finin, and Anupam Joshi (2011). "Generating Linked Data by Inferring the Semantics of Tables". In: *Proceedings of the 1st International Workshop on Searching and Integrating New Web Data Sources - Very Large Data Search*, pp. 17–22.

Ng, Hwee Tou, Chung Yong Lim, and Jessica Li Teng Koo (1999). "Learning to Recognize Tables in Free Text". In: *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, pp. 443–450.

Novelli, Noel and Rosine Cicchetti (2001). "FUN: An Efficient Algorithm for Mining Functional and Embedded Dependencies". In: *Proceedings of the 8th International Conference on Database Theory*, pp. 189–203.

Penn, Gerald, Jianying Hu, Hengbin Luo, and Ryan T. McDonald (2001). "Flexible Web Document Analysis for Delivery to Narrow-bandwidth Devices". In: *Proceedings of the 6th International Conference on Document Analysis and Recognition*, pp. 1074–1078.

Petit, Jean-Marc, Farouk Toumani, Jean-François Boulicaut, and Jacques Kouloumdjian (1996). "Towards the reverse engineering of denormalized relational databases". In: *Proceedings of the 12th International Conference on Data Engineering*. IEEE, pp. 218–227.

Pevzner, Lev and Marti A. Hearst (2002). "A Critique and Improvement of an Evaluation Metric for Text Segmentation". In: *Computational Linguistics* 28.1, pp. 19–36.

Pimplikar, Rakesh and Sunita Sarawagi (2012). "Answering Table Queries on the Web using Column Keywords". In: *Proceedings of the VLDB Endowment* 5.10, pp. 908–919.

Pinto, David, Michael Branstein, Ryan Coleman, W. Bruce Croft, Matthew King, Wei Li, and Xing Wei (2002). "QuASM: A System for Question Answering Using Semi-structured Data". In: *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM, pp. 46–55.

Pinto, David, Andrew McCallum, Xing Wei, and W. Bruce Croft (2003). "Table Extraction Using Conditional Random Fields". In: *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. ACM, pp. 235–242.

Platt, John C. (1998). "Fast Training of Support Vector Machines Using Sequential Minimal Optimization". In: *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Ponte, Jay M. and W. Bruce Croft (1998). "A Language Modeling Approach to Information Retrieval". In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pp. 275–281.

Premerlani, William J. and Michael R. Blaha (1994). "An Approach for Reverse Engineering of Relational Databases". In: *Communications of the ACM* 37.5, 42–ff.

Pyreddy, Pallavi and W. Bruce Croft (1997). "TINTIN: A System for Retrieval in Text Tables". In: *Proceedings of the Second ACM International Conference on Digital Libraries*. ACM, pp. 193–200.

Quercini, Gianluca and Chantal Reynaud (2013). "Entity Discovery and Annotation in Tables". In: *Proceedings of the 16th International Conference on Extending Database Technology*. ACM, pp. 693–704.

Quinlan, John Ross (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.

Rafanelli, Maurizio and Arie Shoshani (1990). "STORM: A Statistical Object Representation Model". In: *Proceedings of the Fifth International Conference on Statistical and Scientific Database Management*, pp. 14–29.

Repici, Dominic John (2015). *HOW-TO: The Comma Separated Value (CSV) File Format*. URL: `http://www.creativyst.com/Doc/Articles/CSV/CSV01.htm`.

Riedl, Martin and Chris Biemann (2012). "TopicTiling: A Text Segmentation Algorithm Based on LDA". In: *Proceedings of ACL 2012 Student Research Workshop*, pp. 37–42.

Robertson, Stephen E., Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, and Mike Gatford (1996). "Okapi at TREC-3". In: *Proceedings of the 3rd Text Retrieval Conference*, pp. 109–126.

Salton, Gerard and Christopher Buckley (1988). "Term-weighting Approaches in Automatic Text Retrieval". In: *Information Processing and Management: an International Journal* 24.5, pp. 513–523.

Sarawagi, Sunita and Soumen Chakrabarti (2014). "Open-domain Quantity Queries on Web Tables: Annotation, Response, and Consensus Models". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 711–720.

Seth, Sharad, Ramana Jandhyala, Mukkai Krishnamoorthy, and George Nagy (2010). "Analysis and Taxonomy of Column Header Categories for Web Tables". In: *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pp. 81–88.

Shoshani, Arie and Harry K. T. Wong (1985). "Statistical and Scientific Database Issues". In: *IEEE Transactions on Software Engineering* 11.10, pp. 1040–1047.

Shoval, Peretz and Nili Shreiber (1993). "Database Reverse Engineering: From the Relational to the Binary Relationship Model". In: *Data and Knowledge Engineering* 10.3, pp. 293–315.

Socher, Richard, John Bauer, Christopher D. Manning, and Andrew Y. Ng (2013). "Parsing with Compositional Vector Grammars". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pp. 455–465.

Son, Jeong-Woo and Seong-Bae Park (2013). "Web Table Discrimination with Composition of Rich Structural and Content Information". In: *Applied Soft Computing* 13.1, pp. 47–57.

Sorrentino, Serena, Sonia Bergamaschi, Maciej Gawinecki, and Laura Po (2009). "Schema Normalization for Improving Schema Matching". In: *Proceedings of the 28th International Conference on Conceptual Modeling*, pp. 280–293.

Spanos, Dimitrios-Emmanuel, Periklis Stavrou, and Nikolas Mitrou (2012). "Bringing Relational Databases into the Semantic Web: A Survey". In: *Semantic Web* 3.2, pp. 169–209.

Stonebraker, Michael, Paul Brown, Jacek Becla, and Donghui Zhang (2013). "SciDB: A Database Management System for Applications with Complex Analytics". In: *Computing in Science and Engineering* 15.3, pp. 54–62.

Suchanek, Fabian M., Gjergji Kasneci, and Gerhard Weikum (2007). "Yago: A Core of Semantic Knowledge". In: *Proceedings of the 16th International Conference on World Wide Web*.

Syed, Zareen and Tim Finin (2011). "Creating and Exploiting a Hybrid Knowledge Base for Linked Data". In: *Agents and Artificial Intelligence*. Vol. 129. Springer Berlin Heidelberg, pp. 3–21.

Syed, Zareen, Tim Finin, Varish Mulwad, and Anupam Joshi (2010). "Exploiting a Web of Semantic Data for Interpreting Tables". In: *Proceedings of the 2nd Web Science Conference*.

Tang, Jian, Zhaoshi Meng, Xuanlong Nguyen, Qiaozhu Mei, and Ming Zhang (2014). "Understanding the Limiting Factors of Topic Modeling via Posterior Contraction Analysis". In: *Proceedings of the 31st International Conference on Machine Learning*, pp. 190–198.

Tengli, Ashwin, Yiming Yang, and Nian Li Ma (2004). "Learning Table Extraction from Examples". In: *Proceedings of the 20th International Conference on Computational Linguistics*.

Tijerino, Yuri A., David W. Embley, Deryle W. Lonsdale, Yihong Ding, and George Nagy (2005). "Towards Ontology Generation from Tables". In: *World Wide Web: Internet and Web Information Systems* 8.3, pp. 261–285.

Tijerino, Yuri A., David W. Embley, Deryle W. Lonsdale, and George Nagy (2003). "Ontology Generation from Tables". In: *Proceedings of the 4th International Conference on Web Information Systems Engineering*, pp. 242–252.

Tsou, Don-Min and Patrick C. Fischer (1982). "Decomposition of a Relation Scheme into Boyce-Codd Normal Form". In: *ACM SIGACT News* 14.3, pp. 23–29.

Vapnik, Vladimir (1982). *Estimation of Dependences Based on Empirical Data*. Springer Series in Statistics. Springer-Verlag New York, Inc.

Venetis, Petros, Alon Y. Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu (2011). "Recovering Semantics of Tables on the Web". In: *Proceedings of the VLDB Endowment* 4.9, pp. 528–538.

Wang, Chu P. and Hartmut H. Wedekind (1975). "Segment Synthesis in Logical Data Base Design". In: *IBM Journal of Research and Development* 19.1, pp. 71–77.

Wang, Daisy Zhe, Xin Luna Dong, Anish Das Sarma, Michael J. Franklin, and Alon Y. Halevy (2009). "Functional Dependency Generation and Applications in Pay-As-You-Go Data Integration Systems". In: *12th International Workshop on the Web and Databases*.

Wang, Jingjing, Haixun Wang, Zhongyuan Wang, and Kenny Q. Zhu (2012). "Understanding Tables on the Web". In: *Proceedings of the 31st International Conference on Conceptual Modeling*, pp. 141–155.

Wang, Xinxin (1996). "Tabular Abstraction, Editing, and Formatting". PhD thesis. University of Waterloo.

Wang, Xuerui, Andrew McCallum, and Xing Wei (2007). "Topical N-Grams: Phrase and Topic Discovery, with an Application to Information Retrieval". In: *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pp. 697–702.

Wang, Yalin and Jianying Hu (2002). "A Machine Learning Based Approach for Table Detection on the Web". In: *Proceedings of the 11th International Conference on World Wide Web*, pp. 242–250.

Wei, Xing and W. Bruce Croft (2006). "LDA-based Document Models for Ad-hoc Retrieval". In: *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pp. 178–185.

Whissell, John S. and Charles L. A. Clarke (2013). "Effective Measures for Inter-document Similarity". In: *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*. ACM, pp. 1361–1370.

Wong, Wilson, Wei Liu, and Mohammed Bennamoun (2012). "Ontology Learning from Text: A Look Back and into the Future". In: *ACM Computing Surveys* 44.4, 20:1–20:36.

Wu, Wentao, Hongsong Li, Haixun Wang, and Kenny Q. Zhu (2012). "Probase: a Probabilistic Taxonomy for Text Understanding". In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 481–492.

Yakout, Mohamed, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri (2012). "InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables". In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 97–108.

Yin, Xiaoxin, Wenzhao Tan, and Chao Liu (2011). "FACTO: A Fact Lookup Engine Based on Web Tables". In: *Proceedings of the 20th International Conference on World Wide Web*, pp. 507–516.

Yoshida, Minoru and Kentaro Torisawa (2001). "A Method to Integrate Tables of the World Wide Web". In: *In Proceedings of the 1st International Workshop on Web Document Analysis*, pp. 31–34.

Zanibbi, Richard, Dorothea Blostein, and James R. Cordy (2004). "A Survey of Table Recognition: Models, Observations, Transformations, and Inferences". In: *International Journal on Document Analysis and Recognition* 7.1, pp. 1–16.

Zhai, Chengxiang and John Lafferty (2004). "A Study of Smoothing Methods for Language Models Applied to Information Retrieval". In: *ACM Transactions on Information Systems* 22.2, pp. 179–214.

Zhang, Meihui and Kaushik Chakrabarti (2013). "InfoGather+: Semantic Matching and Annotation of Numeric and Time-varying Attributes in Web Tables". In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 145–156.

Zobel, Justin and Alistair Moffat (1998). "Exploring the Similarity Space". In: *SIGIR Forum* 32.1, pp. 18–34.

LIST OF FIGURES

1.1 Thesis Outline . . . 15

2.1 Comparison of Document and Database Tables . . . 19
2.2 Representation of Multidimensional Data . . . 22
2.3 Transformation between Document and Database Tables . . . 22
2.4 Table Recovery Process . . . 23

3.1 Sources for Genuine Web Tables . . . 28
3.2 Representation of Tables in Various File Formats . . . 29
3.3 Categorization of Web Tables . . . 30
3.4 Examples of Table Layouts . . . 31
3.5 Most Common Attribute Labels in Web Tables . . . 33
3.6 Distribution of Table Sizes in Dresden Web Table Corpus . . . 34
3.7 Ontology Generation . . . 38
3.8 Ontology Learning in TANGO . . . 38
3.9 Question Answering . . . 39
3.10 Types of Information Gathering . . . 41
3.11 Entity Augmentation . . . 41
3.12 Web Table Understanding Process . . . 44
3.13 Extensions to Web Table Understanding . . . 49

4.1 Example of Layout Table in HTML . . . 53
4.2 Alternatives for Table Classification . . . 57
4.3 Table Section for Local Features . . . 61
4.4 Confusion Matrix . . . 68

5.1 Evaluation of Context Relevance in User Survey . . . 81
5.2 Paragraph Selection Process . . . 82
5.3 TextTiling Algorithm . . . 84
5.4 WindowDiff Evaluation Metric . . . 85
5.5 Parameter Evaluation for Text Segmentation . . . 85
5.6 Matching Words between Table and Context . . . 86
5.7 Plate Notation for Latent Dirichlet Allocation . . . 92
5.8 Evaluation of Alternative Thresholds for Context Selection . . . 95
5.9 Examples of Attribute-specific Context Information . . . 98
5.10 Example of External Knowledge Base . . . 100

5.11 Example of Table and Context . . . 101
5.12 Evaluation of the Quality of Extracted Noun Phrases . . . 106
5.13 Column Search Evaluation . . . 108
5.14 Table Search Evaluation . . . 108

6.1 Example of Multi-concept Table . . . 112
6.2 Coincidental Functional Dependencies . . . 120
6.3 Semantic Normalization Process . . . 123
6.4 Correlation between Neighbouring Columns . . . 126
6.5 Example Table . . . 127
6.6 Reduction of Functional Dependency Candidates . . . 128
6.7 Classification of Inferred Functional Dependencies . . . 129
6.8 Example for Common Concept Labels . . . 132
6.9 Examples of Attribute Dependency Graphs . . . 135
6.10 Selection of Inter-concept Dependencies . . . 135
6.11 Decomposition of Multi-concept Tables . . . 136
6.12 Schema Frequency in ACSDB . . . 138
6.13 Examples of Web Table Schemas . . . 139
6.14 Evaluation of Impact of Threshold θ_p . . . 142
6.15 Impact of θ_p on Normalization Accuracy . . . 144
6.16 Evaluation of Complete Normalization Process . . . 145
6.17 Evaluation of Table Clustering . . . 146

A.1 Example Table (ExA) . . . 154
A.2 Example Table (DxA) . . . 154
A.3 Example Table (AxE) . . . 154
A.4 Example Table (AxD) . . . 154
A.5 Example Table (ExD) . . . 155
A.6 Example Table (DxD) . . . 155
A.7 Example of Attribute/Value Table . . . 155
A.8 Example of Nested Table . . . 155

B.1 Layout Distribution in DWTC . . . 159
B.2 Layout Distribution in Wikipedia Corpus . . . 159

C.1 Example of Context of HTML Table . . . 162
C.2 Example of Context of Open Data Resource . . . 163
C.3 Example of Context of Open Data Dataset . . . 164

LIST OF TABLES

2.1 Table Recognition Research . . . 25
2.2 Table Understanding Research . . . 26

3.1 Size Statistics of Web Table Corpora . . . 35
3.2 Size Statistics of Open Data Platforms . . . 35
3.3 Context and Header Recovery Research . . . 46

4.1 Table Structure Features . . . 59
4.2 Table Structure Variation Features . . . 59
4.3 Global Content Ratio Features . . . 60
4.4 Local Content Ration Features . . . 62
4.5 Distribution of Table and Layout Classes . . . 64
4.6 Selected Features per Classification Task . . . 65
4.7 Evaluation of Table Detection . . . 69
4.8 Evaluation of Layout Identification . . . 70
4.9 Evaluation of Combined Classification . . . 70
4.10 Feature Selection Evaluation . . . 71
4.11 Comparison of Alternative Classification Approaches . . . 72

5.1 Evaluation of Retrieval Functions to Estimate Context Relevance . . . 89
5.2 Evaluation of Text Similarity Measures to Estimate Context Relevance . . . 91
5.3 Evaluation of LDA to Estimate Context Relevance . . . 94
5.4 Context Size and Frequency Statistics . . . 104
5.5 Statistics of Phrases Extracted from Context . . . 105

6.1 Examples of Value Inconsistencies in Web Tables . . . 121
6.2 Entity Column Classification . . . 125
6.3 Test Corpus Statistics . . . 140
6.4 Evaluation of Entity Column Classification . . . 140
6.5 Indicator Weights . . . 143

B.1 DWTC Table Size Statistics . . . 158
B.2 Wikipedia Table Size Statistics . . . 159

D.1 Detailed Evaluation of Retrieval Functions to Estimate Context Relevance (MMR) . . . 166
D.2 Detailed Evaluation of Retrieval Functions to Estimate Context Relevance (MAP) . . . 167
D.3 Detailed Evaluation of Text Similarity Measures to Estimate Context Relevance (MMR) 168

D.4 Detailed Evaluation of Text Similarity Measures to Estimate Context Relevance (MAP) 169
D.5 Detailed Evaluation of LDA to Estimate Context Relevance (MMR) 170
D.6 Detailed Evaluation of LDA to Estimate Context Relevance (MAP) 171

CONFIRMATION

I confirm that I independently prepared the thesis and that I used only the references and auxiliary means indicated in the thesis.

Dresden, October 21, 2015